НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«До захисту допущено»
Науковий керівник кафедри
_____ І.А. Дичка
«___»_____2019 р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки 6.050103 «Програмна інженерія»

на тему: «Програмне забезпечення для предиктивного введення тексту в браузері»

Виконала:
студентка IV курсу, групи КП-51
Довганюк Ліна Олегівна _____

Керівник:
Доцент кафедри ПЗКС, к.т.н., доцент,
Заболотня Т.М. _____

Консультант з нормоконтролю:
Доцент кафедри ПЗКС, к.т.н.,
Онай М.В. _____

Рецензент:
Доцент кафедри ММСА, к.т.н., доцент,
Дідковська М.В. _____

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____

Київ – 2019 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки (програма професійного спрямування) – 6.050103 «Програмна інженерія»

«ЗАТВЕРДЖУЮ»

Науковий керівник кафедри

_____ І.А. Дичка

«___»_____2018 р.

ЗАВДАННЯ
на дипломний проект студентці
Довганюк Ліні Олегівні

1. **Тема проекту** «Програмне забезпечення для предиктивного введення тексту в браузері» затверджена наказом по університету № 1331-С від «22» травня 2019 р.
2. **Термін подання** студентом завершеного проекту: «7» червня 2019 р.
3. **Вихідні дані для дипломного проектування:** див. Технічне завдання.
4. **Перелік задач, які мають бути вирішені:**
   – розробити загальну структуру програмного забезпечення;
   – розробити алгоритми роботи програми;
   – розробити дизайн сторінок та графічних елементів;
   – виконати програмну реалізацію системи відповідно до вимог технічного завдання;
   – виконати тестування програмного проекту.
5. **Перелік обов'язкового ілюстративного матеріалу:**
   – схема потоку даних системи предиктивного введення тексту в браузері (креслення);
   – діаграма діяльності (креслення);
   – логічна структура системи предиктивного введення тексту в браузері (плакат);
   – схема використання системи предиктивного вводу тексту в браузері (плакат).
6. **Консультанти розділів проекту**

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М.В., доцент		

7. **Дата видачі завдання: «**31» жовтня 2018 р.

КАЛЕНДАРНИЙ ПЛАН-ГРАФІК

№ з/п	Назва етапів роботи та питань, які мають бути розроблені відповідно до завдання	Термін виконання	Примітка
1.	Вивчення літератури за тематикою проекту	15.10.2018	
2.	Розроблення та узгодження технічного завдання	19.11.2018	
3.	Розроблення структури програмного забезпечення	03.12.2018	
4.	Підготовка матеріалів першого розділу дипломного проекту	17.12.2018	
5.	Розроблення дизайну системи та графічних елементів	01.02.2019	
6.	Підготовка матеріалів другого розділу дипломного проекту	25.02.2019	
7.	Програмна реалізація дипломного проекту	15.03.2019	
8.	Тестування програмного забезпечення	19.04.2019	
9.	Підготовка матеріалів третього розділу дипломного проекту	03.05.2019	
10.	Підготовка матеріалів четвертого розділу дипломного проекту	15.05.2019	
11.	Оформлення документації дипломного проекту	03.06.2019	

Керівник дипломного проекту _____ Т.М.Заболотня

Студент _____ Л.О. Довганюк

АНОТАЦІЯ

Даний проект присвячений розробленню програмного забезпечення для предиктивного вводу тексту в браузері.

У дипломному проекті виконано порівняльний аналіз існуючих додатків для обміну текстовими повідомленнями, що використовують прогностичну функцію під час текстового набору, а також системи швидких відповідей на повідомлення, проаналізовано алгоритми та методи для реалізації модулю предиктивного вводу тексту, обґрунтовано вибір технологій та допоміжних бібліотек серверної та клієнтської частин для реалізації даної системи прогностичного введення.

Розроблений продукт надає користувачам можливість використання підказок, наданих системою, для полегшення процесу обміну інформацією у браузері. Процес аналізу вхідної інформації та формування можливих варіантів вихідної здійснюється автоматично, відповідно до алгоритму, спроектованого для досягнення найкращого результату при введенні тексту у браузері за критерієм повноти пропонованих варіантів підказок.

У даному дипломному проекті розроблено: архітектуру серверної та клієнтської частини програмного забезпечення, алгоритм аналізу вхідної інформації, алгоритм для досягнення найкращого результату при введенні тексту у браузері за критерієм повноти пропонованих варіантів підказок, а також графічні елементи та дизайн веб-розширення браузеру Chrome.

ABSTRACT

This work is devoted to the development of software for predictive text input in the browser.

During the writing of the diploma project the comparative analysis of existing applications for text message exchanging that use the predictive function during the text set was carried out. The system of quick answers for the messaging also was implemented. The algorithms and methods for the implementation of the module for predictive texting were also analyzed. The choice of technologies and auxiliary libraries of the server and client parts was substantiated.

The developed product allows users to use the clues, provided by the system to facilitate the process of information exchange in the browser.

The process of analysis of input information and the formation of possible variants of the answers is carried out automatically, according to an algorithm designed to achieve the best result for entering the text in the browser according to the criterion of completeness of the proposed variants of prompts.

In this work the next artifacts were developed: the server and client software architecture, the algorithm for analysing input information, the algorithm for achieving the best result for typing in the browser based on the criterion of completeness of the generated variants of prompts, also graphic elements and design of web extension for the Chrome browser.

ДП.045440-01-90 Програмне забезпечення для предиктивного введення тексту в браузері. Відомість проекту

Позначення	Найменування	Кіл-ть	Примітка
	Документація проекту		
ДП.045440-02-91	Програмне забезпечення	4	
	для предиктивного		
	введення тексту в браузері.		
	Технічне завдання		
ДП.045440-03-81	Програмне забезпечення	64	
	для предиктивного		
	введення тексту в браузері.		
	Пояснювальна записка		
ДП.045440-04-51	Програмне забезпечення	4	
	для предиктивного		
	введення тексту в браузері.		
	Програма та методика		
	тестування		
ДП.045440-05-34	Програмне забезпечення	4	
	для предиктивного		
	введення тексту в		
	браузері. Керівництво		
	користувача		

Позначення	Найменування	Кіл-ть	Примітка
ДП.045440-06-99	Програмне забезпечення	1	
	для предиктивного		
	введення тексту в браузері.		
	Потік даних предиктивного		
	введення системи. Схема		
	потоку даних		
ДП.045440-07-99	Програмне забезпечення	1	
	для предиктивного		
	введення тексту в браузері.		
	Діяльність користувача у		
	системі. Схема діаграми		
	діяльності		
ДП.045440-08-98	Програмне забезпечення	1	
	для предиктивного		
	введення тексту в браузері.		
	Компакт-диск		

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2018 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ПРЕДИКТИВНОГО ВВЕДЕННЯ ТЕКСТУ В БРАУЗЕРІ

Технічне завдання

ДП.045440-02-91

"ПОГОДЖЕНО"

Керівник проекту:

_____ Т.М. Заболотня

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ Л.О. Довганюк

2018

ЗМІСТ

НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Програмне забезпечення для предиктивного введення тексту в браузері.

Галузь застосування: інформаційні технології.

1. ПІДСТАВА ДЛЯ РОЗРОБЛЕННЯ

Підставою для розроблення є завдання на дипломне проектування, затверджене кафедрою програмного забезпечення комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» («КПІ ім. Ігоря Сікорського»).

2. ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для використання у веб-браузерах у якості веб-розширення з метою надання користувачам функцій предиктивного введення тексту.

3. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

Програмний продукт повинен забезпечувати такі основні функції:

1) можливість використання у веб-браузері;
2) можливість аналізу вхідних текстових даних;
3) можливість динамічного аналізу введених текстових даних;
4) можливість динамічного отримання підказки у полі введення;
5) можливість налаштування системи.

Додаткові вимоги:

1) наявність візуалізації дій користувача.

4. ВИМОГИ ДО ПРОЕКТНОЇ ДОКУМЕНТАЦІЇ

У процесі виконання проекту повинна бути розроблена наступна документація:

1) пояснювальна записка;

2) програма та методика тестування;

3) керівництво користувача;

4) креслення:

- «Потік даних системи предиктивного введення тексту в браузері. Схема потоку»;
- «Діяльність користувача у системі. Схема діаграми діяльності».

5. ЕТАПИ ПРОЕКТУВАННЯ

Вивчення літератури за тематикою роботи..... 15.10.2018

Розроблення та узгодження технічного завдання 19.11.2018

Розроблення структури програмного забезпечення..... 03.12.2018

Розроблення дизайну сторінок та графічних елементів..... 01.02.2019

Програмна реалізація дипломного проекту 15.03.2019

Тестування програмного забезпечення 19.04.2019

Підготовка матеріалів текстової частини проекту 03.05.2019

Підготовка матеріалів графічної частини проекту..... 30.05.2019

Оформлення технічної документації проекту..... 03.06.2019

6. ПОРЯДОК ТЕСТУВАННЯ РОЗРОБКИ

Тестування розробленого програмного продукту виконується відповідно до «Програми та методики тестування».

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2019 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ПРЕДИКТИВНОГО ВВЕДЕННЯ ТЕКСТУ В БРАУЗЕРІ

Пояснювальна записка

ДП.045440-03-81

"ПОГОДЖЕНО"

Керівник проекту:

_____ Т.М. Заболотня

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ Л.О. Довганюк

2019

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

ПЗ – програмне забезпечення.

Веббраузер – програмне забезпечення, що дає можливість взаємодіяти з елементами на гіпертекстовій веб-сторінці.

Інтелектуальний текст – це технологія введення, що спрощує процес набору тексту, пропонуючи слова, які кінцевий користувач може вставити в текстовому полі.

iOS – IPhone operating system.

Google I/O – щорічна конференція для розробників, яка проводиться компанією Google для обговорення розвитку їх технологій.

T9 – текст на 9 клавіш, система предиктивного введення на малих пристроях.

Python – інтерпретована об’єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією.

Парадигма програмування – система правил для визначення стилю написання програм.

S – мова програмування для написання програм, орієнтованих на вирішення проблем в області статистики.

Scheme – мультипарадигмна мова програмування.

JavaScript (JS) – динамічна, об'єктно-орієнтована прототипна мова програмування.

R – мова програмування і програмне середовище для статистичних обчислень, аналізу та зображення даних у графічному вигляді.

ECMAScript – стандарт мови програмування.

Java — об'єктно-орієнтована мова програмування.

AI – artificial intelligence – штучний інтелект.

ML – machine learning – машинне навчання.

DL – deep learning – глибоке навчання.

ANNs – artificial neural networks – штучні нейронні мережі.

Бекенд – back-end – невидима частина сайту, яка відповідає за роботу сайту.

HTML – це один з основних будівельних блоків будь-якого сайту, диктує організацію і контент сайту.

CSS – це один з основних будівельних блоків будь-якого сайту, містить код для кожного графічного елемента – від фонів до шрифтів – який становить зовнішній вигляд веб-сайту.

Скрипт – програмні інструкції виконання.

MVC – Model View Controller – шаблон контролера модельного перегляду. MTV – Model Template View – вигляд шаблону моделі.

Фреймворк – програмне забезпечення, яке полегшує процес розроблення на об’єднання різних модулів програмного проекту.

ДРНМ – BRNN – bidirectional recurrent neural network – двостороння рекурентна нейронна мережа.

РНМ – RNN – recurrent neural network – рекурентна нейронна мережа.

ВСТУП

Наш час є епохою інформаційних технологій, що пронизують всі сфери діяльності людини. Як наслідок, комунікації всередині суспільства поступово стають переважно цифровими. Серед них і соціальні мережі, і платформи для вільного спілкування на відстані тощо. Не дивлячись на те, що наразі активно розвиваються голосові та відео засоби комунікації між людьми, текстові месенджери не втрачають своєї позиції, адже їх найбільшою перевагою є економія використання трафіку і можливість передачі повідомлень навіть за умов поганого рівня сигналу. Але разом з перевагами існують і недоліки додатків для обміну текстовими даними, найпомітнішими з яких є певна повільність введення тексту (особливо за допомогою смартфону чи мобільного телефону), а також висока ймовірність припущення друкарських помилок або помилок через неграмотність в повідомленнях. Це все обумовлює необхідність розроблення програмних механізмів для предиктивного введення тексту, які мають вдосконалити процес обміну інформацією.

Таким чином, даний дипломний проект присвячено розробленню програмного забезпечення, яке дозволить досягти найкращого результату при введенні тексту у браузері за критерієм повноти пропонованих варіантів підказок.

1. ОГЛЯД ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ

1.1. Огляд проблеми, яка вирішується ПЗ

Впродовж останніх десятиліть проблема обміну інформації стала чи не найактуальнішою серед людей. Як результат було створено програмні застосунки для вирішення питання комунікації. Дані рішення мають ряд переваг, серед яких: зручність використання, автоматизація процесу обміну інформацією та коректна робота. Але разом з тим вони мають і недоліки такі, як: повільність ведення тексту, ймовірність припущення друкарських помилок або помилок через неграмотність в повідомленнях. Внаслідок аналізу існуючих програмних механізмів, що використовують прогностичну функцію під час текстового набору, було виявлено, що вони не в повній мірі можуть вирішувати задачу підтримки предиктивного введення тексту за критерієм повноти множини підказок, що пропонуються. Тому головною проблемою є відсутність автоматизованої системи предиктивного введення тексту, а саме в браузері, яка б дозволила отримати необхідний результат за вказаним критерієм. Вона у свою чергу є наслідком інших проблем, таких як: використання великої кількості часу для написання тексту, негативного впливу людських факторів (неграмотності), великої ймовірності виникнення помилки при швидкому наборі тексту, зниження продуктивності написання та ін. Випливаючи з цього, метою даної роботи є створення програмного рішення для автоматизованого предиктивного введення тексту в браузері.

1.2. Опис вимог до розроблюваного ПЗ

Сформовано ряд вимог, яким має відповідати програмне рішення предиктивного введення тексту в браузері. Перш за все, створюваний застосунок має володіти словником тієї чи іншої мови. При взаємодії користувача з продуктом головна увага приділяється словам, які формують текст для відправлення. Тому наявність словника дасть можливість

перевірити на коректність введене слово. По-друге, для зменшення можливих комбінацій букв при наборі тексту важливим є аналіз його складових. Необхідно враховувати частоту використання слів, їх можливі комбінації з іншими словами, тобто використовувані користувачем словосполучення, та їх морфологічну форму. Також для реалізації інтуїтивно-зрозумілого інтерфейсу застосунку важливою є організація графічного представлення дій користувача, а саме: виводу тексту світлішого тону на фон, візуалізації рухів курсору по клавіатурі з відображенням відповідного слова, яке при цьому утворюється або просто виводу можливого слова в окрему область на екрані та ін.

Таким чином, можна узагальнити вищезгадані вимоги:

1. Наявність словника тієї чи іншої мови.
2. Підтримка аналізу використовуваних користувачем слів та їх комбінацій.
3. Інтуїтивно-зрозумілий інтерфейс.
4. Візуалізація дій користувача.
5. Використання у веб-браузері.

Отже, розглянуто зазначені вимоги, згідно з якими проведено детальний порівняльний аналіз існуючих програмних рішень. Також враховано їх переваги та недоліки при розробленні програмного застосунку.

1.3. Аналіз існуючих рішень

При розгляді питання про створення системи для полегшення процесу написання повідомлень було встановлено, що найбільш популярним рішенням є використання інтелектуального аналізу тексту [1]. Це технологія введення, що спрощує процес набору тексту, пропонуючи слова, які кінцевий користувач може вставити в текстовому полі. Прогноз ґрунтується на контексті інших слів у повідомленні та вписаних перших букв. Оскільки кінцевий користувач просто натискає на слово, а не набирає його на

клавіатурі, предиктивний процес введення тексту може значно прискорити процес.

Таким чином, проаналізовано програмні засоби, розроблені відомими компаніями.

1.3.1. Аналіз програмного рішення, розробленого компанією Apple

Дане програмне рішення включило прогностичну функцію текстового блоку під назвою QuickType [2] у випуску iOS 8 (див. рис. 1.1). QuickType має компонент машинного навчання, який дозволяє створювати власні словники. Це допомагає програмному забезпеченню запам'ятовувати такі речі, як використовуваний кінцевим користувачем сленг при спілкуванні із конкретними людьми та відповідно коригує його можливі текстові складові.

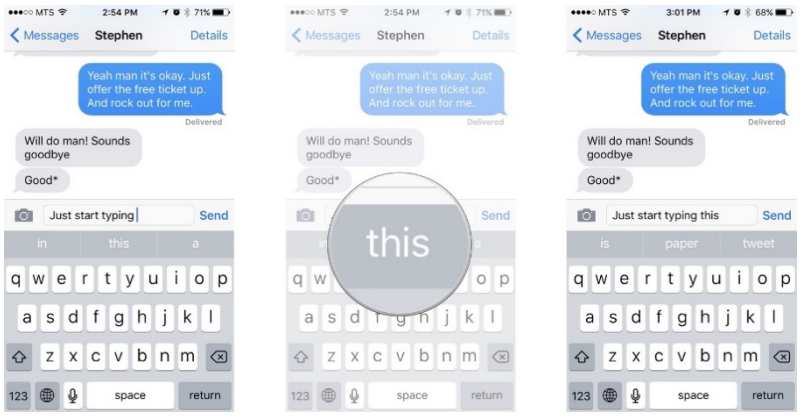

Рис. 1.1. Ілюстрація роботи прогностичної функції текстового набору, розробленої компанією Apple

Незважаючи на те, що інтелектуальні текстові технології Apple стають все більш витонченими, програмне забезпечення, як відомо, схильне до помилок. Наприклад, при написанні повідомлень при використанні інтелектуального аналізу введеного тексту на пристроях Apple іноді називається ефектом Купертино. Ім'я було пов’язане з тим, що початкові перевірки вписаних даних заміняли слово «cooperation»

(з англ. «кооперація», «співпраця» та ін.) на «Cupertino», яке є просто містом в Каліфорнії, де Apple має штаб-квартиру компанії.

Із врахуванням визначених раніше вимог для даного програмного рішення виділено наступне: перевагами системи є наявність словника, який формується на основі вживаних користувачем слів, його аналіз та коригування текстових складових повідомлення відповідно до нього, візуалізація запропонованих варіантів написання слова для використання під час переписки та наявність інтуїтивно-зрозумілого інтерфейсу. При цьому слід зазначити, що є ймовірність виникнення помилок, пов’язаних із неправильним функціонуванням системи, наприклад, пропонування слів, які не відповідають тематиці бесіди.

1.3.2. Аналіз програмного рішення, розробленого компанією Android

Після оголошення про випуск Jelly Bean 4.1 [3], який передбачав впровадження предиктивного текстового введення, у 2012 було детально вивчено деякі нові функції останньої версії Android. Серед безлічі функцій особливе місце займало додавання інтелектуального аналізу тексту. На відміну від більшості програмних клавіатур, які просто виправляли окремі слова під час введення. Google надав клавіатурі можливість вивчати особливості друку користувача. Це означало, що при частішому використанні система ставало більш розумною, тим самим краще прогнозувала текст, який набирався. Через деякий час до вже існуючих функцій було підібрано найбільш часто використовувані фрази та короткі запитання. У результаті подальших покращень Android разом з Google мають систему інтелектуального введення тексту, представлену на рис. 1.2.

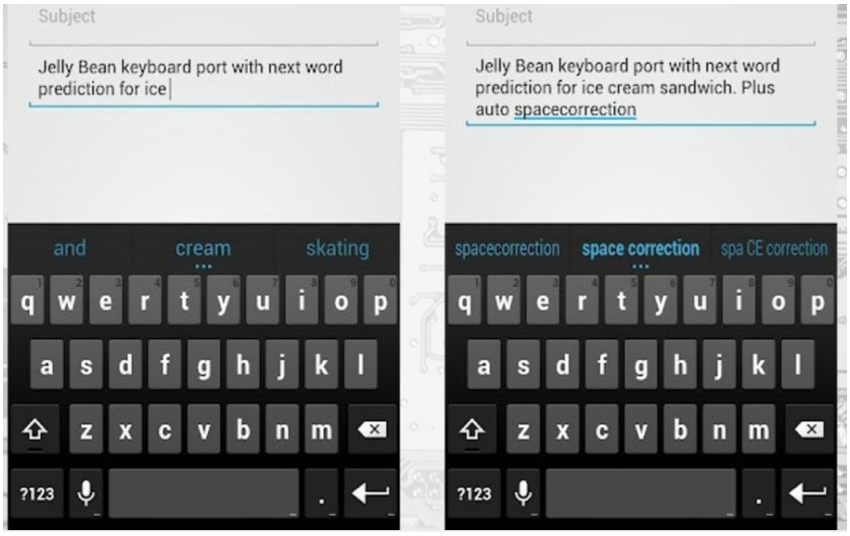

Рис. 1.2. Приклад роботи прогностичної функції текстового набору, розробленої компаніями Android та Google

Відповідно до вимог, описаних вище, проаналізовано дане програмне рішення. Щодо переваг, то система має архів слів, які використовуються користувачем найчастіше і при подальшій взаємодії проводиться коригування введених даних. Слід відмітити наявність інтуїтивно зрозумілого інтерфейсу, який представлений у вигляді розширення клавіатури. Воно являє собою окремий простір із можливим варіантом слова. Серед недоліків слід виділити наступне: часткові випадки пропонування слів, які не зовсім відповідають тематиці повідомлення

1.3.3. Аналіз програмного рішення, розробленого компанією Gmail

У травні 2018-го року під час щорічної конференції розробників Google-I/O [4] було представлено розширення поштового сервісу Gmail. Серед основних нововведень – Smart Compose (з англ. – «розумне доповнення»). Воно є доповненням до вже відомої функції Smart Reply (з англ. – «розумна відповідь»), яка аналізувала зміст листа за допомогою нейронної мережі. Основною метою впровадженого програмного рішення є написання електронних листів за користувача. Йому достатньо лише почати

друкувати, як система почне передбачати можливе закінчення повідомлення.

Із виступу представників компанії Google стало відомо, що зазначена функція працює у фоновому режимі і пропонує користувачеві ті чи інші фрази, які більше підходять за змістом до заданого контексту (див. рис. 1.3). Таким чином, при використанні даної системи, прогнозується зменшення ймовірності допущення граматичної чи орфографічної помилки. Також було зазначено, що згадана функція розуміє контекст та пропонує слова, аналізуючи тематику та текст повідомлення. Можна відзначити, що станом на 2018-2019 роки система працює лише у веб-версії сервісу Gmail, оскільки вона є експериментальною функцією.

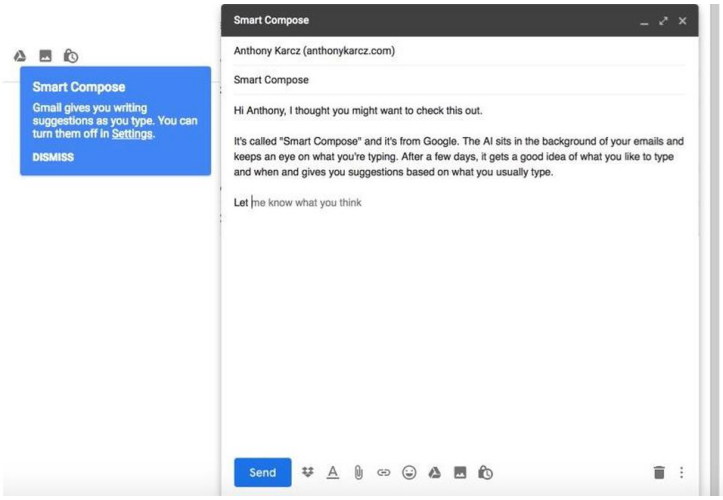

Рис. 1.3. Ілюстрація роботи прогностичної функції текстового набору сервісу Gmail

З точки зору обраних вимог для аналізу програмного рішення, дана система має ряд переваг, а саме: словник користувача, тобто перелік найчастіше вживаних ним слів, візуалізацію можливих закінчень речення, аналіз контексту повідомлень та інтуїтивно-зрозумілий інтерфейс для взаємодії із системою. Але слід враховувати і те, що текстові складові доступні тільки англійською мовою а також, що програмне рішення реалізовано лише для використання веб-версії у сервісі Gmail.

1.3.4. Аналіз програмного рішення, використовуваного в LinkedIn

Окрім згаданих вище функцій предиктивного введення тексту, слід відзначити систему швидких відповідей на повідомлення, яка впроваджена у сервісі LinkedIn [6]. Вона являє собою короткі попередньо сформовані слова, які користувач може використати як шаблон для написання реакції на отриманий лист. Система надає декілька варіантів можливого тексту. Вони розташовані у нижній частині екрану. Найпоширенішими у використанні є: «Дякую», «Не впевнений». Після натискання на відповідний варіант, швидка відповідь відправляється автоматично. Якщо ж користувачеві необхідно написати власний текст, то для цього йому необхідно звернутися до спеціально відведеного поля. При використанні даної системи слід пам’ятати, що параметри швидкої відповіді залежать від типу повідомлення, яке було отримано. При цьому, вони можуть використовуватися лише тоді, коли користувач повідомляє іншого користувача. Інтерфейс програмного рішення показаний на рис. 1.4.

Після огляду функції швидких відповідей згаданого сервісу проаналізовано її за вказаними вище вимогами. Перевагою є наявність слів, які представлені у вигляді шаблонів відповідей, тобто існування певного словника. Якщо розглядати взаємодію з користувачем, то система є досить зрозумілою для використання. Недоліками ж програмного рішення є обмеженість у словах, тобто для формування відповіді доступні лише деякі фрази, а сама система не надає можливості інтерактивного доповнення тексту під час написання власного повідомлення.

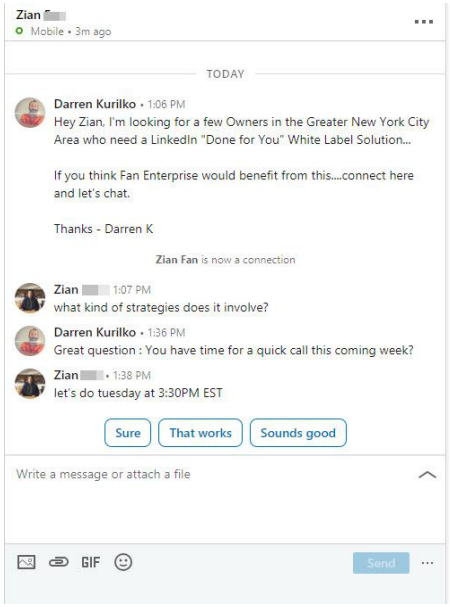

Рис. 1.4. Приклад роботи системи швидких повідомлень сервісу LinkedIn

1.3.5. Аналіз програмного рішення T9

Одним з перших додатків для інтелектуального написання тексту є T9 (текст на 9 клавіш) [6]. Система полегшує процес друкування на мобільних телефонах та інших малих пристроях шляхом зменшення багатократного натискання, що вимагало від користувача набагато більше зусиль. При цьому букви об’єднуються в групи на кожній клавіатурі телефону, комбінація яких формує необхідний текст. Програмне забезпечення в пристрої коригує послідовність натиснених клавіш, тим самим порівнює вхідні дані із вже відомими системі словами та пріоритетні прогнози за частотою використання. Ілюстрацію процесу взаємодії з даною системою подано на рис. 1.5.

Проаналізувавши програмне рішення відповідно до вимог, можна виділити наступне: система має інтуїтивно-зрозумілий інтерфейс для взаємодії з користувачем, існує словник вживаних слів, які пропонуються при вводі перших літер відповідного текстового компоненту. Недоліком є те, що дана функція доповнення застосовується лише для лексем, а не тексту в цілому.

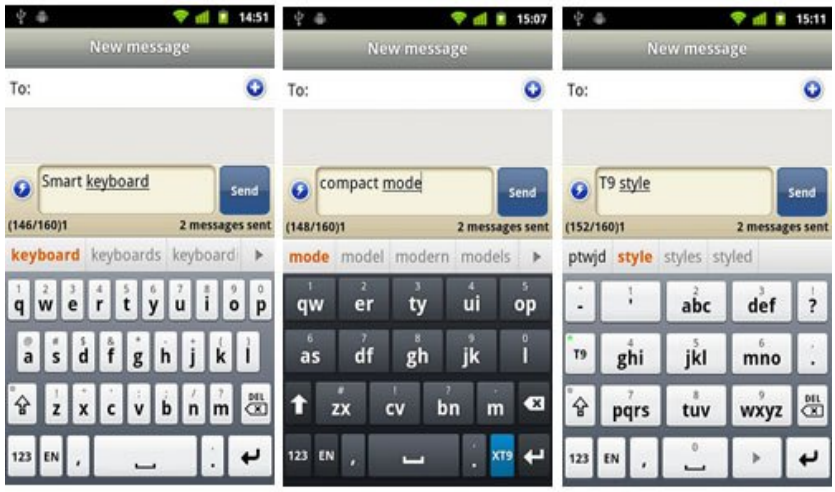

Рис. 1.5. Візуалізація роботи прогностичної функції текстового набору сервісу T9

1.4. Результати проведеного аналізу

Виходячи з отриманих даних після проведення аналізу, можна стверджувати, що згадані системи повністю не вирішують задачу предиктивного введення тексту в браузері. Більшість із описаних програм орієнтовані на використання у телефонах або інших малих пристроях, а ті програмні рішення, котрі призначені для веб-браузерів, мають певні недоліки, які впливають на процес взаємодії з користувачем.

Отже, сформовано порівняльну таблицю результатів аналізу програмних рішень (див. табл. 1.1).

Таким чином, для програмного рішення предиктивного введення тексту в браузері виділено ряд вимог:

1. Система має володіти словником відповідної мови.
2. Необхідна наявність аналізу введених даних та змісту отриманих повідомлень.
3. Необхідна наявність візуалізації дій користувача для інтуїтивного процесу взаємодії системи.
4. Орієнтованість на використання програмного засобу у веб-браузері.

Таблиця 1.1

Результати аналізу існуючих програмних рішень

Критерії / Програмні рішення	Наявність словника	Аналіз слів та їх комбінацій	Візуалізація дій	Використання у веб-браузері
Apple	+	+/–	+	–
Android	+	+/–	+	–
Gmail	+	+	+	+
LinkedIn	+/–	–	+	+
T9	+	+/–	+	–

Також програмне рішення має вирішувати наступні проблеми:

1. Використання великої кількості часу при написанні тексту.
2. Негативний вплив людського фактору, а саме повільне написання та неграмотність.
3. Некоректне оброблення великих об’ємів вхідної інформації.
4. Зниження продуктивності написання повідомлень.

Отже, як результат після проведення аналізу існуючих рішень відповідно до вимог маємо наступні завдання для створюваного програмного застосунку:

1. Розроблення алгоритму пошуку поля вводу.
2. Розроблення алгоритму для аналізу вхідної інформації:
   2.1. Пошук вхідних повідомлень.
   2.2. Пошук ключових слів.
   2.3. Аналіз відповідей.

3. Створення модуля для аналізу даних.
4. Забезпечення візуалізації дій користувача при роботі з програмою.
5. Створення програмного застосунку у вигляді розширення для браузера.
6. Реалізація інтуїтивно-зрозумілого інтерфейсу користувача.

2. ОБГРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1. Вибір мови програмування для розроблення

Відповідно до поставленої задачі, а саме створення програмного застосунку для предиктивного введення тексту в браузері, випливає необхідність вибору оптимальних засобів розроблення. Оскільки програмне рішення базується на аналізі та використанні текстових даних, тому є сенс виокремити певні вимоги до використовуваних інструментів.

Вимоги до мови програмування:

- підтримка функціональної парадигми [7];
- наявність бібліотек для аналізу даних;
- наявність високого рівня продуктивності за критерієм обсягу використаних даних та витраченого часу.

2.1.1. Мова програмування Python

Python – інтерпретована об’єктно-орієнтована мова програмування високого рівня. Має строгу динамічну типізацію [8]. Також має високорівневі структури даних, що дозволяють забезпечити швидкий процес розроблення програм. Завдяки підтримці модулів та пакетів модулів Python дозволяє повторно використовувати код. Стандартні бібліотеки та інтерпретатор даної мови є доступними як у скомпільованій формі, так і у вихідній на всіх основних платформах. Також слід зазначити підтримку кількох парадигм програмування, зокрема:

- об’єктно-орієнтованої;
- процедурної;
- функціональної;
- аспектно-орієнтованої.

Отже, Python є досить популярною мовою програмування загального призначення, яка широко застосовується для обробки даних.

Виходячи з цього, маємо наступні переваги:

1. Популярна мова програмування та має багато розширень та підтримку спільноти розробників.
2. Підтримує функціональну парадигму.
3. Має простий та зрозумілий синтаксис.
4. Пакети, такі як pandas, scikit-learn і Tensorflow, надають Python перевагу при розробленні сучасних застосунків з машинним навчанням.

Серед недоліків слід виділити наступне:

1. Python – мова з динамічною типізацією.

2.1.2. Мова програмування R

R – мова програмування і програмне середовище для статистичних обчислень, аналізу та зображення даних у графічному вигляді [9]. Відомо, що створення даної мови програмування має за основу синтаксис мови програмування S [10] із семантикою, успадкованою від Scheme [11]. Але при цьому, незважаючи на деякі принципові відмінності, більшість програм, написаних мовою програмування S, запускаються в середовищі R.

Також мову програмування R зазвичай використовують для здійснення статистичних аналізів, включаючи лінійну і нелінійну регресію, класичні статистичні тести, аналіз часових рядів (серій), кластерний аналіз тощо, оскільки вона має значні можливості для цього. Слід зазначити, що завдяки використанню додаткових функцій і пакетів доступних на сайті Comprehensive R Archive Network (CRAN), R легко розбудовується. Більша частина стандартних функцій, написана мовою R, однак при цьому є можливість використання коду, написаного на C, C++ або Фортраном. Також за допомогою програмного коду на C або Java можна безпосередньо маніпулювати R-об'єктами.

Виходячи з цього, маємо наступні переваги:

1. Доступні статистичні функції та методи.

2. Гарна якість візуалізації даних при використанні бібліотек, таких як ggplot2.
3. Велика кількість спеціалізованих пакетів для статистичного застосунку, наприклад, нейронної мережі, нелінійної регресії, тощо.

Серед недоліків слід виділити наступне:

1. Погана продуктивність.
2. Використовується лише для статистики та обробки даних.
3. Відмінність від інших мов програмування (індексація починається з «1», декілька операторів присвоєння, інша структура даних).

2.1.3. Мова програмування Javascript

JavaScript (JS) – динамічна, об'єктно-орієнтована прототипна мова програмування [12]. Відома як реалізація стандарту ECMAScript [13]. Зазвичай використовують для створення сценаріїв веб-сторінок, оскільки вона надає можливість на стороні клієнта (пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд веб-сторінки.

JavaScript можна віднести до прототипних, скриптових мов програмування з динамічною типізацією. Окрім прототипної, JavaScript також частково підтримує інші парадигми програмування (наприклад, імперативну та частково функціональну) і деякі відповідні архітектурні властивості, зокрема:

- динамічну та слабку типізацію;
- автоматичне керування пам'яттю;
- прототипне наслідування;
- викоритання функцій як об'єктів першого класу.

Виходячи з цього, маємо наступні переваги:

1. Наявність асинхронного вводу/виводу.

2. Після появи Node.js [14] є найчастіше використовуваною мовою програмування для написання серверів.

Серед недоліків слід виділити наступне:

1. Має невелику кількість бібліотек для аналізу великого об’єму даних.

2.1.4. Мова програмування Java

Java – об’єкто-орієнтована мова прогамування [15]. Відповідно до офіційної реалізації, Java-програми компілюються у байт-код. При його виконанні, завдяки віртуальній машині, він інтерпретується для конкретної платформи.

Розроблення даної мови мало за основу синтаксис із C і C++. Зокрема, взято за основу об'єктну модель C++, проте її модифіковано. При цьому, враховано та усунуто можливість появі конфліктних ситуацій, що могли виникнути через помилки програміста. Також полегшено сам процес розроблення об'єктно-орієнтованих програм. Ряд дій, які в C/C++ зазвичай виконують програмісти, виконується віртуальною машиною. Передусім Java розроблялась як платформо-незалежна мова, тому вона має менше низькорівневих можливостей для роботи з апаратним забезпеченням, що в порівнянні, наприклад, з C++ зменшує швидкість роботи програм. За необхідності таких дій Java дозволяє викликати підпрограми, написані іншими мовами програмування.

Виходячи з цього, маємо наступні переваги:

1. Популярна мова програмування.
2. Строго типізована мова.
3. Високопродуктивна компільована мова програмування загального призначення.
4. Може бути використана для написання бізнес-логіки та для аналітики великих обсягів даних.

Серед недоліків слід виділити наступне:

1. Не зовсім зрозумілий синтаксис для спеціалізованого аналізу.
2. Невелика кількість бібліотек для роботи зі статистикою.

Таким чином, після огляду наведених вище мов програмування можемо стверджувати, що майже усі розглянуті мови задовольняють зазначені вимоги. Оскільки R має значні можливості для здійснення статистичних аналізів, вона може бути використана як засіб реалізації програмного рішення. Python та Java також можуть обробляти та працювати з великим обсягом даних. Усі мови програмування, окрім Java, підтримують функціональну парадигму. Також у кожної наявні бібліотеки для роботи та аналізу даних великих розмірів. Узагальнене порівняння розглянутих мов програмування відповідно до зазначених вимог представлено у вигляді табл. 2.1.

Таблиця 2.1

Порівняння мов програмування для реалізації предиктивного вводу тексту

Мова програму-вання	Підтримка функціональ-ної парадигми	Наявність бібліотек для обробки та аналізу даних	Продуктивність при роботі з великими об’ємами даних	Зручність використання
Python	+	+	+	+
R	+	+	–	+/–
JavaScript	+	+/–	–	+/–
Java	–	+	+	+/–

За результатами проведеного аналізу зроблено висновок, що всі зазначені мови програмування майже повністю задовольняють встановлені вимоги. Для обраної теми дипломної роботи основним критерієм вибору є

продуктивність при роботі з великим обсягом даних. Також важливими є зручність використання для отримання бажаного кінцевого результату. Тому серед усіх виділених мов програмування обрано Python.

2.2. Вибір технологій для проведення аналізу даних

Розроблення програмного застосунку для предиктивного введення тексту в браузері має за основу оброблення великих обсягів даних. Для коректної реалізації аналізу має сенс використання відповідних засобів реалізації, спрямованих на роботу з текстовими даними.

2.2.1. Штучний інтелект або AI

Як випливає з назви, штучний інтелект можна легко інтерпретувати як певне «вживлення» людського інтелекту до машин.

Штучний інтелект або AI – це більш широка концепція. Вона включає в себе технології, починаючи від старої AI (GOFAI) до таких як глибоке навчання [16]. Тобто машина може виконувати завдання, що ґрунтуються на певному наборі встановлених правил. Вони у свою чергу вирішують проблеми або певні алгоритми. Саме такий процес поведінки і являє собою те, що називається штучним інтелектом. Наприклад, такі машини можуть переміщати і маніпулювати об'єктами, фіксувати рух об'єктів, їх положення та його зміну або вирішити інші проблеми.

AI-машини, як правило, можна класифікувати на дві групи – загальні і вузькі. Загальні штучні інтелектуальні AI машини можуть розумно вирішувати проблеми, як ті, що наведені вище. Вузькі, у свою чергу, можуть виконувати конкретні завдання, іноді навіть краще, ніж люди – хоча вони мають обмежений обсяг даних, з яким працюють. Технологія, яка використовується для класифікації зображень у Pinterest [17], є прикладом вузького AI.

2.2.2. Машинне навчання або Machine Learning

Як випливає з назви, машинне навчання, далі ML, може бути вільно інтерпретоване як поняття розширення можливостей комп'ютерних систем із можливістю «вчитися» [16]. Насправді, машинне навчання являє собою використання алгоритмів для аналізу даних, навчання з них, а потім визначення або прогнозування певної події. Таким чином, замість використання ручного кодування програмного забезпечення з певним набором інструкцій для виконання конкретної задачі, застосовують машинне навчання, використовуючи великі обсяги даних і алгоритмів, які дають їй можливість навчитися самостійно виконувати завдання.

Алгоритмічні підходи протягом багатьох років включали навчання дерева рішень, індуктивне логічне програмування, кластеризацію та байєсівські мережі. На жаль, жодна з них не досягла кінцевої мети загального AI.

Як відомо, ML є підмножиною штучного інтелекту і насправді це просто спосіб реалізації AI – це метод алгоритмів навчання, який дозволяє навчитися приймати рішення. На практиці, щоб дізнатися більше про оброблювану інформацію, надається велика кількість даних алгоритму для навчання в ML.

Як виявилося, однією з найкращих прикладних областей для машинного навчання протягом багатьох років був комп'ютерний зір, хоча це все ще вимагало великої кількості ручного кодування для виконання роботи. Тобто люди писали б вручну класифікатори, такі як фільтри виявлення країв, щоб програма могла визначити, де об'єкт починається і закінчився; виявлення форми, щоб визначити, скільки вона сторін; класифікатор для розпізнавання букв певних написів. З усіх цих написаних вручну кодуючих класифікаторів можна було б розробили алгоритми, щоб зрозуміти зображення і «навчитися» визначати призначення даного напису.

2.2.3. Глибоке навчання або Deep Learning

Як відомо, глибоке навчання, далі DL, є підмножиною ML [16]. Насправді, це просто техніка реалізації машинного навчання, або іншими словами, є черговою еволюцією машинного навчання.

Подібно до того, як люди використовують мозок, щоб ідентифікувати моделі та класифікувати різні типи інформації, алгоритми глибокого навчання можна навчити виконувати ті ж завдання для машин. Але, на відміну від біологічного мозку, де будь-який нейрон може з'єднатися з будь-яким іншим нейроном на певній фізичній відстані, ці штучні нейронні мережі мають дискретні шари, з'єднання та напрямки поширення даних.

Мозок зазвичай намагається розшифрувати отриману інформацію. Це досягається завдяки маркуванню та призначенню елементів у різні категорії. Кожен раз, коли отримано нову інформацію, мозок намагається порівняти її з відомим елементом, перш ніж вивчити його – це однакова концепція глибоких алгоритмів навчання. Наприклад, штучні нейронні мережі (artificial neural networks або ANNs) [18] є типом алгоритмів, які мають на меті наслідувати спосіб прийняття рішень нашими мізками.

Таким чином, машинне навчання стрімко розвивається. Це дає можливість вибору різних підходів до вирішення тих чи інших завдань. Якщо ж порівнювати глубоке навчання та машинне навчання, то перше із зазначених може автоматично виявляти функції, які будуть використовуватися для класифікації. А ML у свою чергу вимагає, щоб ці функції були задані вручну. Крім того, на відміну від машинного навчання, DL потребує високотехнологічних машин і значно більшої кількості тренувальних даних для отримання точних результатів.

Отже, для даного дипломного проекту обрано глибоке навчання, або DL. На думку автора, це є ефективним засобом реалізації поставлених задач з точки зору якості отриманих результатів та витраченого часу.

2.3. Вибір мови програмування для клієнтської частини

Як відомо, складовими усіх веб-сайтів є три компоненти, а саме: база даних, серверна та клієнтська частина. Останній з них – просто браузер, який використовується людиною для перегляду сайту. Клієнт може знаходитися у будь-якому місці, де користувач може переглядати сайт: мобільні пристрої, настільні комп'ютери або ноутбуки. У свою чергу, сервер може знаходитися у віддаленому місці у будь-якій точці світу, що зберігає дані про місцезнаходження, при цьому керує внутрішньою архітектурою сайту, обробляє запити та відправляє сторінки на браузер. Таким чином, серверний скрипт виконується веб-сервером, а клієнтський сценарій запускається браузером.

Програмні інструкції на стороні клієнта вбудовані в код розмітки HTML [20] на веб-сайті, який розміщується на сервері мовою сумісним із браузером або скомпільований для зв'язку з ним. Це і є клієнтською мовою програмування.

Відомо, що виконання інструкцій на клієнті завжди розвивається. Воно стає більш простим, швидшим та зручнішим у використанні. У результаті сайти також стають швидшими, ефективнішими за критерієм відношення об'єму виконаної роботи до затраченого часу та при цьому на сервері зменшується кількість роботи, яку необхідно виконати.

Розроблення бекенда в проектах веб-застосунків в основному полягає у написанні компонентів, які користувачі не бачать (підключення до бази даних, системи кешу, продуктивність, безпека та транзакції, тощо). Але для взаємодії з ними необхідно створювати відповідний інтерфейс.

Програмні інструкції виконання впроваджуються всередині коду сторінки і взаємодіють з HTML сайту, вибираючи його елементи, а потім маніпулюють цими елементами, щоб забезпечити інтерактивний процес. Скрипти взаємодіють з каскадним файлом таблиці стилів (CSS [20]), який стилізує відображення сторінки. Він диктує, яку роботу повинен виконати

серверний код, і повертає дані, які повертаються з бекенд у вигляді, що читається браузером.

Таким чином сформовано широке уявлення про те, що таке інтерфейсна технологія та розглянуто деякі з найбільш широко використовуваних мов сценаріїв і інтерфейсних фреймворків для реалізації клієнтської частини дипломного проекту.

Мови майже завжди використовуються в контексті їх фреймворків, які швидко виконують складний код з бібліотеками спільно використовуваного коду і великої кількості надбудов. Розробник може використовувати одну або їх комбінацію при створенні фронтенду сайту.

2.3.1. JavaScript

JavaScript – це клієнтська скриптова мова програмування. Найбільш широко використовуваний клієнтський скрипт. Зазвичай інтерфейс сайту – це поєднання JavaScript, HTML і CSS. Для JavaScript існує безліч відмінних фреймворків, які спрощують його і надають йому більшу гнучкість, наприклад:

1. AngularJS [21] – надійне середовище JavaScript для сайтів з «важкими» даними.
2. JQuery [22] – швидка, невелика бібліотека об’єктів JavaScript, яка спрощує управління JavaScript в різних браузерах.
3. Bootstrap [23] – платформа для мобільних пристроїв, що використовує HTML, CSS і JavaScript для прискорення розробки швидкодіючих додатків React, для дизайну користувальницького інтерфейсу Express.js, Backbone.js, Ember.js, MeteorJS і інші.
4. AJAX (JavaScript + XML) [24] – технологія, що дозволяє оновлювати певні частини сайту без поновлення повної сторінки шляхом асинхронного підключення до бази даних і витягування фрагментів даних на основі JSON або XML.

5. React [25] – це гнучка та ефективна бібліотека JavaScript для створення інтерфейсу користувача, і вона вважається найбільш затребуваною на ринках у 2018 році. У моделі MVC React відповідає за складову представлення або View, які у свою чергу є файлами без логіки, якими керує контролер. React не замінює View, скоріше, робить цю складову більш модульною, створюючи компоненти, що багаторазово використовуються (списки, сортувальні таблиці тощо).

2.3.2. PHP

PHP (Hypertext PreProcessor, препроцесор гіпертексту) – мова програмування, що виконується на стороні веб-сервера, спроектований) в якості інструменту створення динамічних та інтерактивних веб-сайтів [26]. Ця мова виявилася досить гнучкою і потужною, тому набула великої популярності і використовується в проектах будь-якого масштабу: від простого блогу до найбільших веб-додатків в Інтернеті. Серед фреймворків можна виділити наступне [27]:

1. Lavarel – фреймворк, який має структуру з елегантним синтаксисом, дозволяє розширити базову функціональність, має інтеграцію зі сторонніми бібліотеками.
2. CodeIgniter – це фреймворк PHP, який використовує архітектуру Model View Controller (MVC). Досить простий для розуміння та має відмінну документацію.
3. Symfony – володіє дуже гнучким фреймфворком. Він включає в себе пакетну і компонентну систему, яка дозволяє вибирати потрібні функції PHP або просто використовувати всю інфраструктуру.
4. Zend – це об'єктно-орієнтована, заснована на MVC інфраструктура, яка дозволяє завантажувати тільки необхідні компоненти, як окремі бібліотеки.

Таким чином, після аналізу описаних вище засобів реалізації клієнтської частини зроблено висновки щодо використання їх у розробленні програмного застосунку: кожна із зазначених мов програмування є непоганим рішенням для виконання поставлених перед розробником користувацького інтерфейсу завдань. Але з точки зору витраченого часу на вивчення та освоєння основ мови програмування для реалізації клієнтської частини, JavaScript дозволить виконати поставлені задачі більш ефективно.

3. СТРУКТУРНО-АЛГОРИТМІЧНА ОРГАНІЗАЦІЯ

3.1. Вимоги до програмного забезпечення

З огляду на поставлене завдання реалізації системи предиктивного вводу тексту в браузері виникає необхідність у визначенні та проведенні аналізу вимог до розроблюваного програмного забезпечення. Оскільки, для чіткого розуміння задач розроблення програмного продукту слід дослідити проблематику розглянутої сфери та сформувати відповідні цілі, які мають бути досягнуті в кінці розроблення, описано проблеми, які вирішує даний проект.

3.1.1. Аналіз проблематики та цілей системи предиктивного вводу

Головною проблемою на думку автора є затрати часу при формуванні вихідного повідомлення. У свою чергу вона є результатом інших проблем, а саме:

1. Проблема швидкого аналізу текстових даних. При написанні повідомлення витрачається багато часу на аналіз вхідних даних та формування вихідних даних.
2. Велика ймовірність виникнення помилки. При написанні повідомлення є велика ймовірність зробити помилку та не помітити її при обробці великих об’ємів використовуваних даних. Також є вплив людського фактору, а точніше, неграмотності.
3. Проблема ефективності використання системи. З точки зору відношення використаного часу та швидкості отримання кінцевого результату, використання незручних систем призводить до зниження продуктивності користувача.

Якщо об’єднати зазначене вище, матимемо дерево проблем представлене на рис. 3.1.

З отриманих даних після дослідження проблематики сформовано цілі, які мають бути досягнуті у кінцевому результаті розроблення продукту. Загальною метою проекту є реалізація розширення для браузерів, яке матиме змогу пропонувати альтернативні закінчення речень для користувачів.

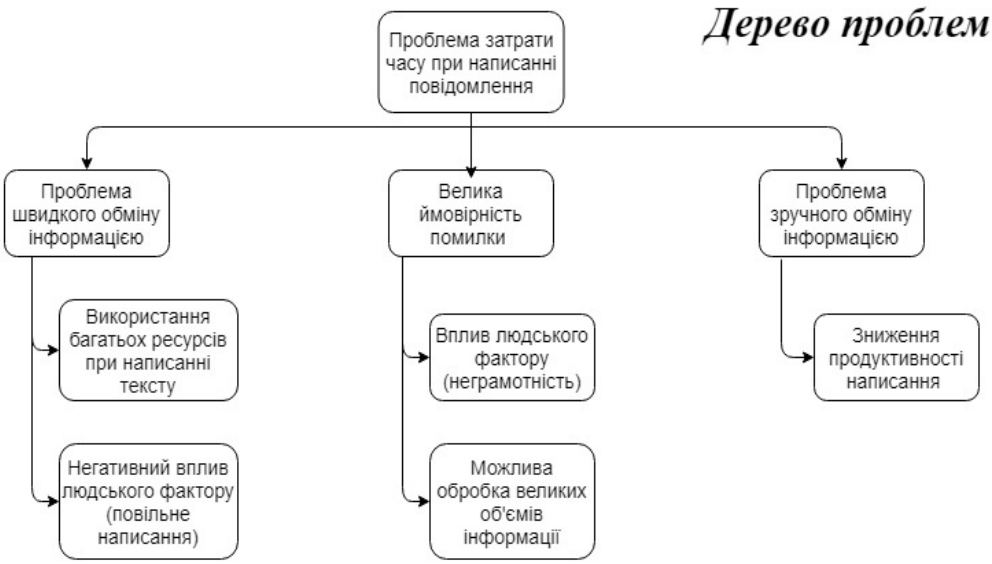

Рис. 3.1. Ілюстрація структури дерева проблем, які вирішує розроблюване програмне забезпечення

Також доцільною є реалізація у такому програмному забезпеченні специфічних функціональних характеристик як, наприклад: володіння системи словником певної мови; підтримка аналізу введених даних та вмісту отриманих повідомлень для виділення певної тематики розмови.

Важливою характеристикою є можливість візуальної диференціації дій користувача та підказок системи (наприклад, відображати згенеровані підказки більш світлим відтінком шрифту) для підвищення рівня інтуїтивної зрозумілості інтерфейсу користувача. Така реалізація дозволить зменшити час на написання тексту, при цьому зменшити ймовірність виникнення помилок, покращити умови використання системи предиктивного введення. Загальна структура цілей продукту зображена у вигляді дерева цілей (див. рис. 3.2).

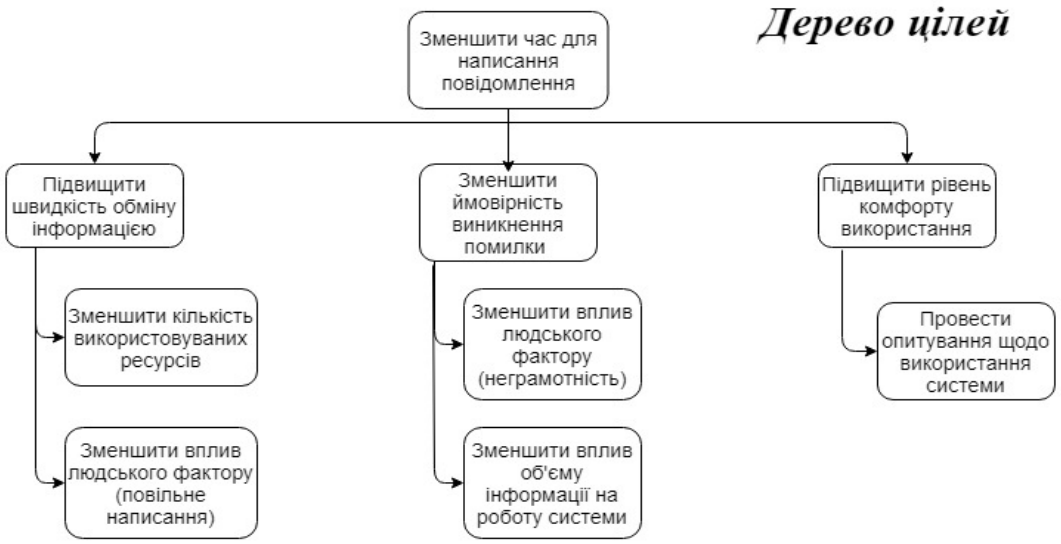

Рис. 3.2. Ілюстрація структури дерева цілей продукту для розроблюваного програмного забезпечення

Взявши за основу отримані цілі, сформовано дерево результатів, які мають бути отримані в процесі розроблення продукту (див. рис. 3.3).

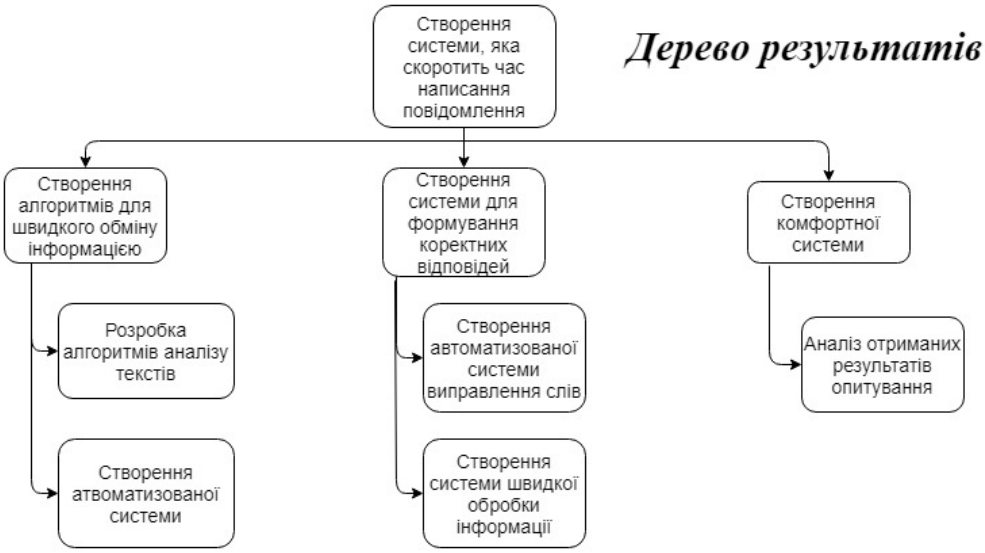

Рис. 3.3. Дерево результатів продукту

Отже, провівши дослідження проблематики та сформувавши цілі та очікувані результати продукту, сформовано ряд вимог, яким має відповідати розроблюване програмне забезпечення.

3.1.2. Аналіз вимог до програмного забезпечення предиктивного вводу

Відповідно до стандарту IEEE Standard Glossary of Software Engineering Terminology [28] поняття «вимоги» можна визначити як:

1. Умови або можливості, які необхідні користувачеві для досягнення поставлених цілей або вирішення проблем.
2. Умови або можливості, якими має володіти система або її компоненти для того, щоб відповідати стандартам або виконати контракт.
3. Документоване представлення можливостей для пунктів 1 та 2.

Відповідно до класичного технічного підходу вимоги використовуються на стадії проектування ПЗ, але вони також можуть бути використані в процесі перевірки ПЗ, оскільки тестування системи базується на раніше описаних вимогах.

Фаза розробки вимог може бути розбита на такі етапи:

1. Виявлення вимог – збір, розуміння, розгляд та з'ясування потреб зацікавлених сторін проекту.
2. Аналіз вимог – перевірка цілісності та завершеності вимог.
3. Специфікація – документування вимог.
4. Перевірка правильності.

Вимоги до програмного забезпечення складаються з трьох рівнів, а саме:

1. Бізнес-вимоги – визначають призначення програмного забезпечення, зазвичай описуються в документах про бачення (англ. vision) та границях проекта (англ. scope).
2. Вимоги користувача – визначають набір задач користувача, які мають бути виконані програмою, а також способи або ж сценарії їх рішення в системі. Зазвичай дані вимоги можуть бути представлені у вигляді способів використання (англ. use cases), історій користувача (англ. user story) та сценаріїв взаємодії (англ. scenario).

3. Функціональні вимоги – визначають як реалізувати продукт. Описується в системній специфікації (англ. system requirement specification, SRS [29]).

Також розглянуто якість вимог розроблюваного програмного забезпечення. Їх характеристика описана у вигляді таблиці (див. табл. 3.1).

Таблиця 3.1

Характеристика якості вимог ПЗ

Характеристика	Опис
Одиничність	Вимога описує одну і тільки одну річ.
Завершеність	Вимога повністю визначена в одному місці, наявна уся необхідна інформація
Послідовність	Вимога не суперечить іншим вимогам і повністю відповідає зовнішній документації.
Атомарність	Вимога «атомарна», тобто вона не може бути розбита на більш детальні вимоги без втрати завершеності.
Актуальність	Вимога не стала застаріла протягом часу.
Недвозначність	Вимога коротко визначена без звернення до технічного жаргону, акронімам та іншим прихованим формулюванням; вона виражає об’єктивні факти, не суб’єктивну думку; визначення не містить нечітких фраз.

Продовження табл. 3.1

Характеристика	Опис
Виконуваність	Вимога може бути реалізована у рамках проекту.
Обов’язковість	Вимога представляє характеристику зацікавленої сторони, відсутність якої призведе до неповноцінності рішення; вона не може бути проігнорована.
Верифікованість	Реалізація вимоги може бути визначена за допомогою одного з можливих методів: огляд, демонстрація, тест чи аналіз.

Таким чином, для визначення конкретних вимог до розроблюваного дипломного проекту є необхідність у виокремленні зацікавлених сторін проекту.

Стейкголдери, також зацікавлені сторони (від англ. stakeholders) – фізичні та юридичні особи, які мають легітимний інтерес у діяльності організації, тобто певною мірою залежать від неї або можуть впливати на її діяльність [29]. Вони забезпечують можливість виокремлення необхідних для створення функцій системи та є джерелом вимог до неї.

Для програмного забезпечення предиктивного вводу тексту в браузері зацікавленими сторонами є користувачі веб-браузерів та розробники системи.

Таким чином, після проведення аналізу вимог до програмного забезпечення виокремлено ряд вимог, яким має відповідати система. Усі вони були сформовані в результаті проведення опитування користувачів

подібних систем. Це дало можливість зробити акцент на необхідності розробки саме тих характеристик, які очікуються (див. табл. 3.2).

Таблиця 3.2

Вимоги до системи предиктивного вводу тексту

Код вимоги	Опис вимоги
Вимоги до інтерфейсу	Зрозуміла візуалізація дій користувача у системі.
Вимоги до продуктивності	Система має працювати стабільно та швидко.
Вимоги до безпеки використання даних	Дані мають бути захищені від можливого вилучення сторонніми особами.
Вимоги до функціоналу	Створення необхідного функціоналу у повному обсязі.
Вимоги до підтримки програми	Система має підтримуватися та вдосконалюватися. Створення нової версії програми.
Вимоги до портативності	Система має бути доступною у різних браузерах.
Вимоги до доповнення тексту	Динамічне отримання підказок.
Вимоги до змісту доповнюваного тексту	Зміст написаного повідомлення має базуватися на щоденній переписці, із урахуванням можливих умов її написання: бізнес-розмова, розмова з друзями, тощо.

3.2. Логічна структура розроблюваної системи

Отже, провівши дослідження проблематики та сформувавши цілі та результати продукту, описано логічну структуру програмної системи предиктивного вводу тексту в браузері.

Основною метою побудови логічної структури є специфікація набору об’єктів, за рахунок використання яких буде реалізована програмна

структура системи. При цьому корисну роль грають діаграми, що дозволяють створити візуальне відображення структури проекту, підвищуючи загальне розуміння системи.

Розглядаючи програмне забезпечення в цілому, виділено основні модулі (див. Додаток 1):

- модуль навченої моделі нейронної мережі;
- модуль взаємодії з моделлю нейронної мережі;
- модуль синхронізації даних між веб-розширенням та сервером;
- модуль логіки веб-розширення;
- модуль взаємодії з користувачем.

Таким чином, маємо логічну структуру розроблюваного програмного забезпечення для предиктивного введення тексту в браузері. Взаємодія між модулями являє собою передачу необхідної інформації з модуля обробки та генерування даних у випадку запиту зі сторони модуля представлення даних при взаємодії з користувачем і навпаки (див. Додаток 1).

3.3. Модуль оброблення та генерування даних

За основу для створення модулю оброблення та генерування даних для предиктивного введення тексту запропоновано взяти широко розповсюджені на даний момент нейронні мережі, які використовуються для вирішення складних задач, що потребують аналітичних обчислень, подібних до тих, які робить людський мозок. Як відомо, найбільш поширеними прикладами використання нейронних мереж є класифікація, розпізнавання та передбачення певних даних, у тому числі і текстових. Тому використання нейронної мережі дозволить підвищити повноту варіантів підказок, що пропонуються програмою, зменшити необхідність у ручному налаштуванні моделі предиктивного введення, яке може призвести до отримання некоректних даних, що для задачі моделювання мови є неприпустимим.

Нейромережеві технології, які пропонуються для використання у даній роботі, це архітектура seq2seq [31], BRNN [32] та Beam Search [33].

Базова sequence-to-sequence модель, складається з двох рекурентних нейронних мереж (RNN): encoder (кодер), яка обробляє вхідні дані, і decoder (декодер), яка генерує дані виведення (див. рис. 3.5).

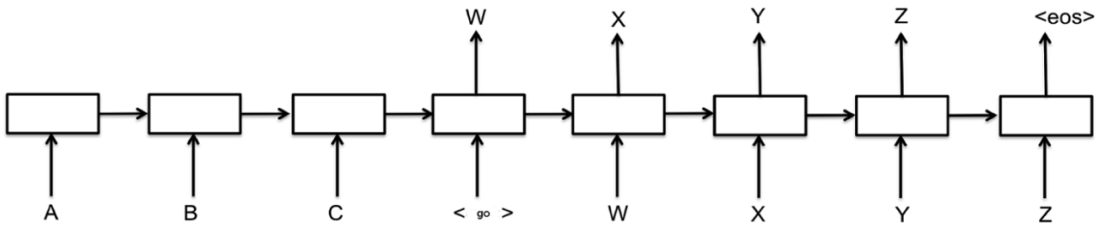

Рис. 3.5. Зображення базової архітектури sequence-to-sequence моделі

У відповідності до рисунку дані вводу кодера представлені буквами *A*, *B*, *C*, а дані вводу декодера – *GO*, *W*, *X*, *Y*, *Z*.

Кожен прямокутник на зображенні є кліткою в RNN, зазвичай кліткою GRU – керованого поворотного блоку, або кліткою LSTM – довгої короткостроковій пам'яті. Кодери і декодери можуть мати загальні ваги або ж, частіше, використовувати різні набори параметрів.

У базовій моделі кожне введення повинно бути закодоване у вектор стану фіксованого розміру, оскільки це єдине, що передається декодеру. Це є найсерйознішим недоліком простої архітектури кодер-декодер. Цей підхід є не зовсім вдалим для поставленої задачі, зважаючи на те, як комп'ютери зазвичай вирішують задачу стиснення. Коли стискається файл за допомогою архіватора, розмір результуючого файлу приблизно пропорційний розміру вихідного файлу. Але це не зовсім вірно, оскільки розмір стисненого файлу пропорційний кількості інформації в початковому файлі, а не його розміру. Проте було припущено, що розмір вихідного файлу досить точно відповідає кількості інформації в ньому.

Продовжуючи аналогію з комп'ютерами, збережено всі речення не у формі вектора фіксованого розміру, а у формі, що містить кількість банків

даних, яка дорівнює кількості вихідних слів. Це реалізується за допомогою двосторонньої рекурентної нейронної мережі, що складається з прямої РНМ (forward RNN) і зворотньої РНМ (backward RNN). Як і випливає з їх назв, пряма і зворотна РНМ зчитують вихідне речення в прямому і зворотньому напрямку відповідно.

Як відомо, РНМ узагальнює послідовність, зчитуючи по одному елементу за раз. Це означає, що пряма РНМ узагальнює вихідне речення до *j*-го слова, починаючи з першого слова, а зворотня РНМ – до *j*-го слова, починаючи з останнього. Тобто разом узагальнюють все вхідне речення (див. рис. 3.6).

Однак узагальнення на позиції кожного слова не є ідеальним узагальненням всього вхідного речення. Завдяки своїй послідовності РНМ краще пам'ятає останні слова. Тобто, чим далі вхідне слово знаходиться від *j*, тим менш імовірно, що прихований стан РНМ добре його пам'ятає. Анотаційний вектор (англ. annotation vector) найкращим чином представляє поточне слово $\overrightarrow{w_i}$.

Тому розглянуто анотаційний вектор, як контекстно-залежне уявлення слова (англ. context-dependent word representation). Більш того, можемо трактувати набір контекстно-залежних уявлень, як механізм, за допомогою якого збережено вихідне речення у вигляді подання змінної довжини, на відміну від узагальнення фіксованої довжини в простій моделі кодер-декодер.

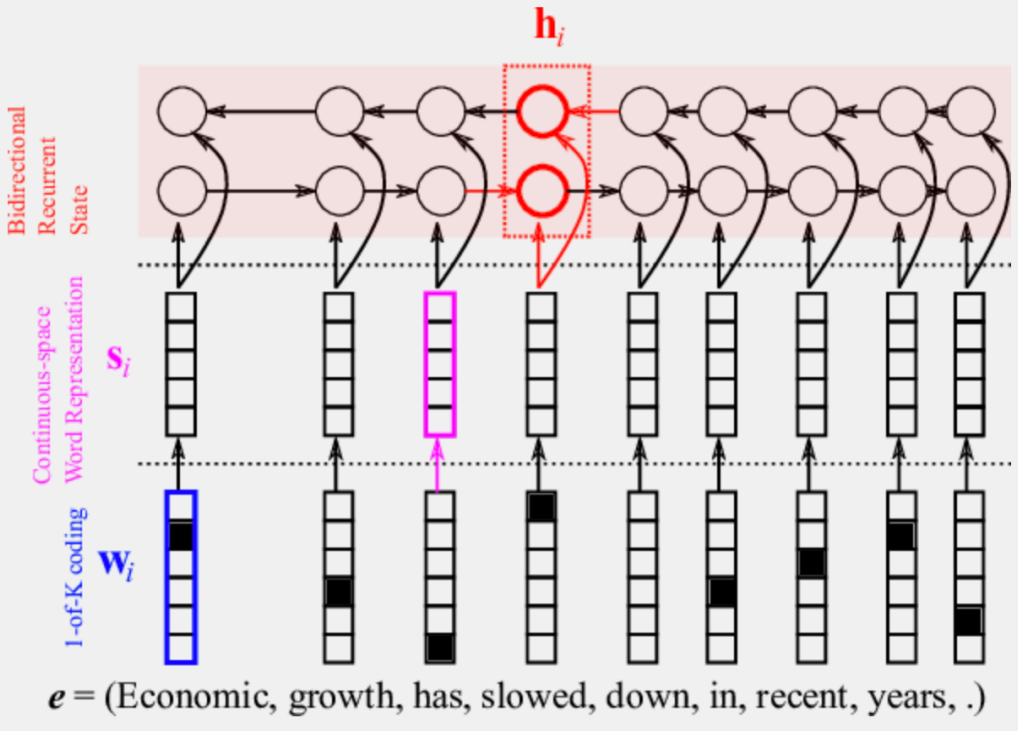

Рис. 3.6. Ілюстрація використання двосторонньої рекурентної мережі для кодування вхідного речення

За наявності подання змінної довжини вихідного речення декодер повинен мати можливість вибірково фокусування на одному або декількох контекстно-залежних уявленнях слів (анотаційних векторах) для кожного цільового слова. Тому визначено, на якому анотаційному векторі повинен фокусуватися декодер в кожен момент часу.

Уявимо, що аналізується деяке вихідне речення, вже написано i–1 перших цільових слів ($y_1, y_2, \ldots, y_{i-1}$) і маємо написати i-е цільове слово.

Зазвичай аналізатор дивиться на кожне вихідне слово x_i (у випадку алгоритму – це контекстно-залежне уявлення h_i), розглядає його спільно з уже проаналізованими словами ($y_1, y_2, \ldots, y_{i-1}$) і вирішує, чи було початкове слово x_i вже проаналізоване. Це еквівалентно тому, наскільки релевантне або нерелевантне вихідне слово x_i кожному цільовому слову. Цей процес повторюється для кожного слова у вихідному реченні.

Також запропоновано включити в декодер невелику нейронну мережу, що реалізує процес аналогічний описаному вище. Ця невелика

нейронна мережа, яку називають «механізмом уваги» [34] (позначена (1) на рис. 3.7), приймає на вході попередній прихований стан декодера $\overrightarrow{z_{i-1}}$ (тобто те, що вже було проаналізовано) і одне з контекстно-залежних уявлень вихідних слів h_l.

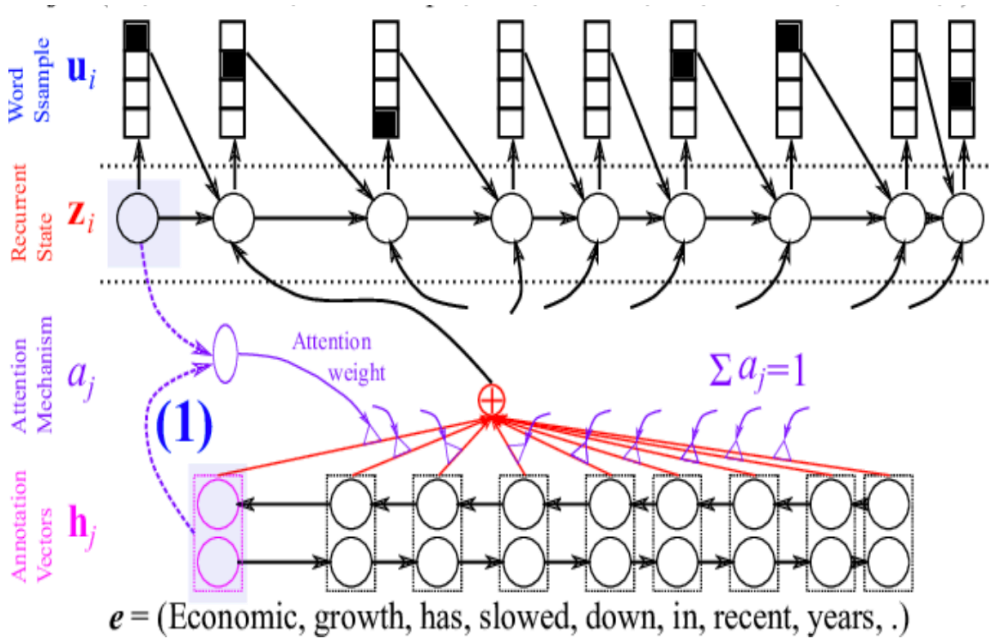

Рис. 3.7. Ілюстрація роботи механізму уваги. Врахування того, що було проаналізовано та вихідного слова

Механізм уваги реалізований у вигляді нейронної мережі з одним прихованим шаром і одним скалярним виходом, як показано на рис. 3.8. Процес повторюється для кожного вихідного слова.

Після того, як розраховано оцінки релевантності для кожного вихідного слова, проведено таке перетворення, щоб сума оцінок дорівнювала одиниці (див. рис. 3.9).

Це зроблено за допомогою софтмакс-нормалізації (softmax normalization [35]) наступним чином:

$$\alpha_j = \frac{\exp(e_j)}{\Sigma_{j'} \exp(e_{j'})}, \tag{3.1}$$

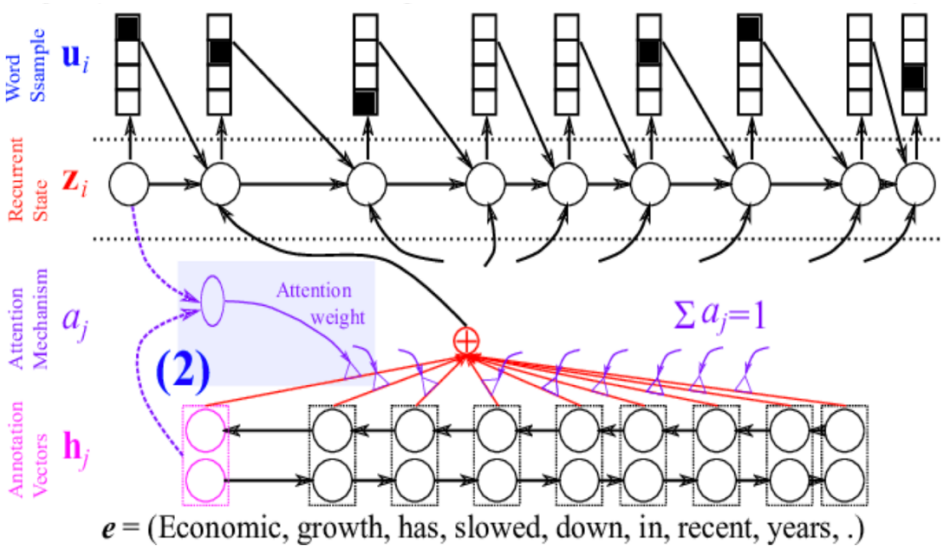

Рис. 3.8. Зображення роботи механізму уваги. Повернення скаляру, що відповідає релевантному значенню *j*-го вихідного слова

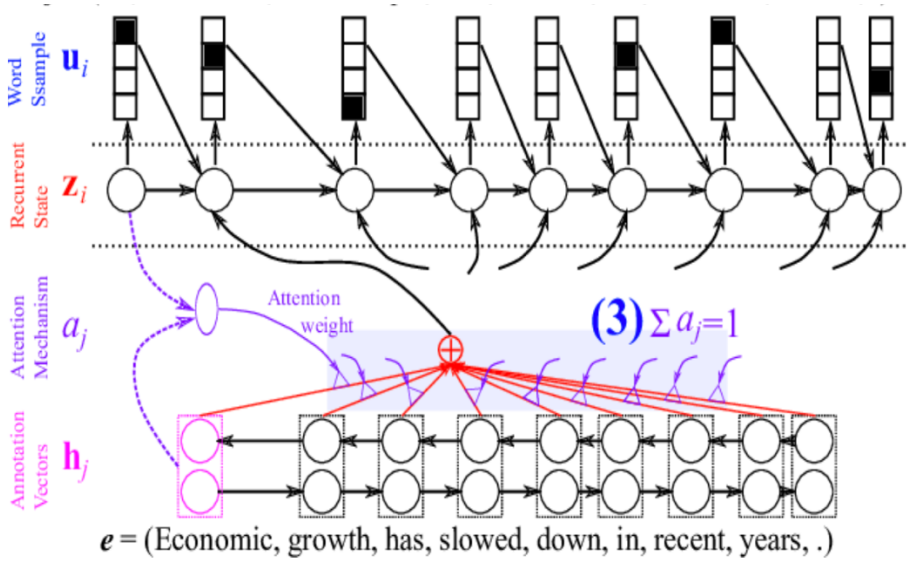

Рис. 3.9. Зображення роботи механізму уваги. Оцінка релевантності, повернутої механізмом уваги

Можна назвати багато причин, чому потрібен подібний тип нормалізації, але найважливіша з них полягає в тому, що це дає можливість інтерпретувати оцінки, присвоєні механізмом уваги, як ймовірності. З точки зору ймовірності, розглянуто вагу уваги (attention weight) α_i, як ймовірність того, що декодер вибере j-е контекстно-залежне уявлення вихідного слова

з усіх *T*-вихідних слів. Потім обчислено очікуване контекстно-залежне уявлення слова в рамках даного розподілу (заданого вагами уваги α_i) наступним чином:

$$c_i = \Sigma_{j=1}^{T} \alpha_j h_j \,, \tag{3.2}$$

Цей очікуваний вектор $\overrightarrow{c_i}$ підсумовує інформацію про все вихідне речення, однак, з різним акцентом на різних місцях/словах вихідного речення. Будь-який вектор анотацій (контекстно-залежний вектор), який вважається відповідним (іншими словами, з великою вагою уваги) механізмом уваги, буде краще представлений, ніж той, що має низьку вагу уваги.

Коли вектор $\overrightarrow{c_i}$ обчислений, далі все відбувається так само, як і було в декодері. Для цього розглянуто архітектуру кодер-декодер більш детально.

Архітектура кодер-декор є основою нейронного машинного перекладу. Фактично, ця архітектура є ядром глибокого навчання, де найбільший акцент робиться на навчанні хорошого уявлення. У певному сенсі, завжди можна розділити будь-яку нейронну мережу навпіл і назвати одну частину кодером, а іншу декодером.

З часу публікації роботи Калчбреннера (Kalchbrenner) і Блансома (Blunsom) в Оксфордському університеті в 2013 році, архітектура кодер-декодер була запропонована багатьма науковими групами, включаючи лабораторію машинного навчання Монреальського університету і компанію Google, як новий підхід до статистичному машинного перекладу [36].

Не дивлячись на те, що і кодер, і декодер можуть бути реалізовані на основі будь-яких типів нейронних мереж, використано РНМ в обох випадках.

Почнемо з кодера, прямого застосування рекурентної нейронної мережі, заснованого на її властивості підсумовування послідовності. Тобто

застосовано рекурентну функцію активації рекурсивно над вхідною послідовністю, або реченням, до кінця, коли кінцевим внутрішнім станом RNN $\overrightarrow{c_T}$ є резюме всього вхідного речення.

Спочатку, кожне слово вихідного речення представляється у вигляді вектора в прямому унітарному коді (one-hot vector або 1-of-K coded vector), як показано на рис. 3.10. Всі слова рівновіддалені один від одного. Тобто жодні взаємозв'язки між словами не зберігаються.

Прийнято ієрархічний підхід до вилучення репрезентативного представлення, вектора, який узагальнює вхідне речення. У цій ієрархії першим кроком є отримання змістовного представлення кожного слова. Тобто таким чином моделі дозволено вивчати дані.

Кодер лінійно проектує вектор у прямому унітарному коді $\overrightarrow{w_i}$ (див. рис. 3.10). Це відбувається за допомогою матриці E, кількість стовпців в якій дорівнює кількості слів у вихідному словнику, а кількість рядків – довільно, зазвичай від 100 до 500. Ця проекція $\overrightarrow{s_i} = E\overrightarrow{w_i}$, представлена на рис. 3.11.

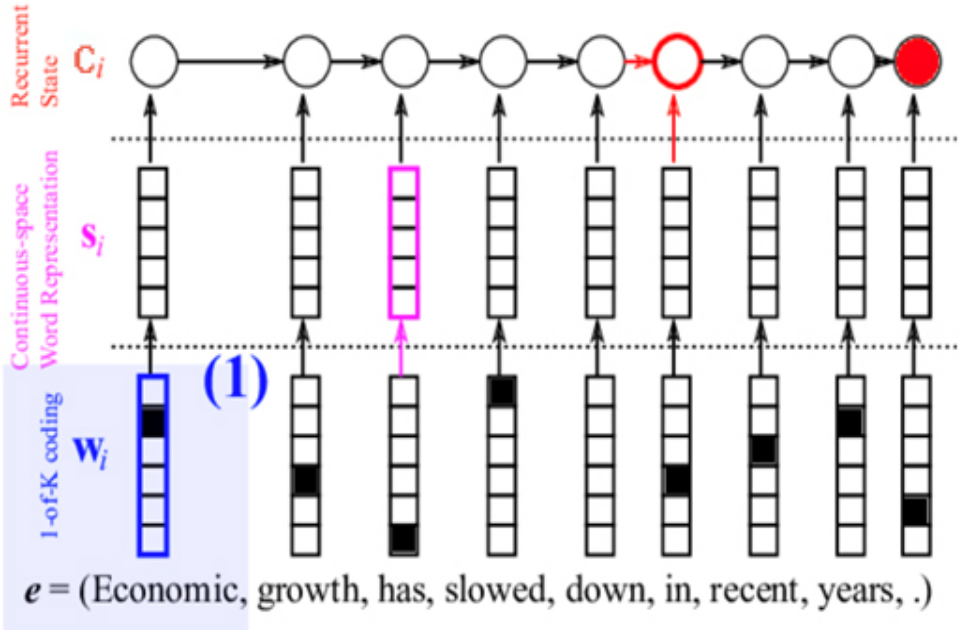

Рис. 3.10. Ілюстрація представлення слова вихідного речення у вигляді вектора в прямому унітарному коді

Вона дає в результаті безперервний вектор (англ. continuous vector) для кожного вихідного слова. Потім кожен елемент вектора оновлюється, щоб максимізувати ефективність аналізу.

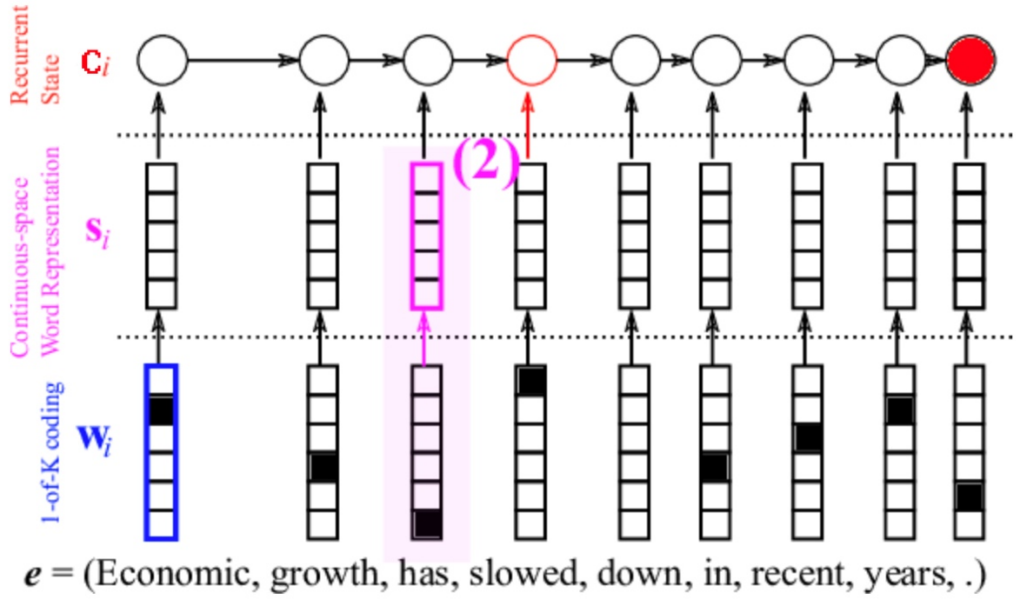

Рис. 3.11. Приклад отримання безперервного вектору для кожного вихідного слова

На даному етапі вже перетворено послідовність слів в послідовність безперервних векторів $\overrightarrow{s_i}$ та використано РНМ.

Однією з двох ключових можливостей РНМ є узагальнення послідовностей. В даному випадку використано РНМ, щоб узагальнити послідовність безперервних векторів, відповідних словами вихідного речення. На рис. 3.12 показано, як РНС виконує це завдання.

Математично процес узагальнення можна виразити таким чином:

$$c_i = \varphi_\theta\left(c_{i-1}, s_i\right), \tag{3.3}$$

де $\overrightarrow{c_0}$ – нульовий вектор. Тобто після того, як останній безперервний вектор $\overrightarrow{s_T}$, що відповідає останньому слову, буде прочитаний, внутрішній стан РНМ $\overrightarrow{c_T}$ буде являти собою узагальнення всього вихідного речення.

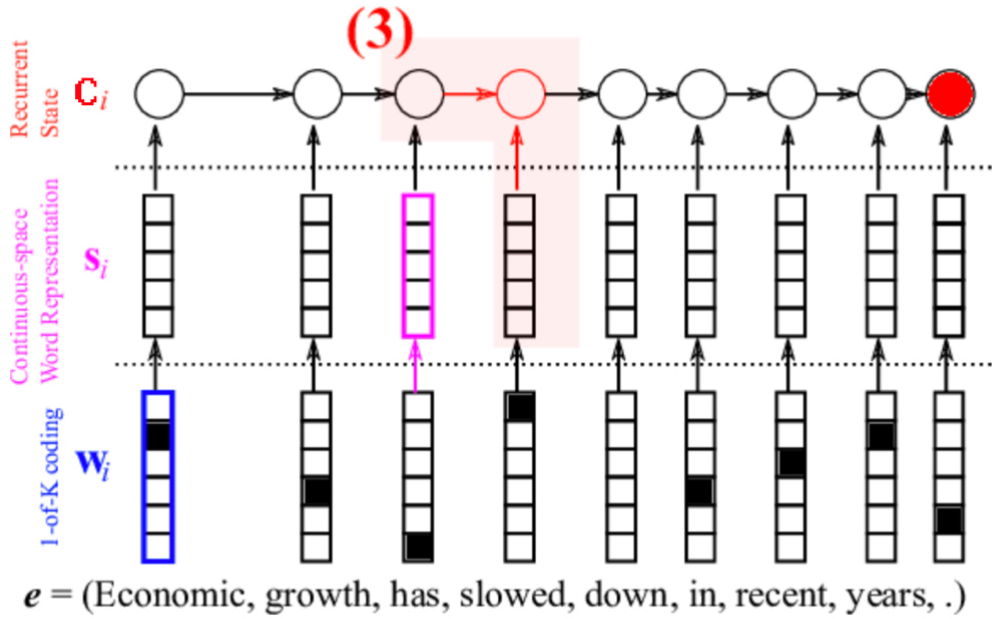

Рис. 3.12. Зображення узагальнення послідовності безперервних векторів РНМ

Отже, на даному етапі сформоване гарне уявлення вихідного речення. Таким чином, виникає необхідність у створенні декодеру. Запропоновано реалізувати даний елемент на основі рекурентної нейронної мережі або РНМ.

Також на даному етапі корисно пам'ятати, що декодер – це, по суті, перевернутий кодер.

Почнемо з обчислення внутрішнього стану РНМ z_i на основі узагальненого вектора вихідного речення $\overrightarrow{c_T}$, попереднього слова u_{i-1} і попереднього внутрішнього стану z_{i-1}. Новий внутрішній стан z_i обчислено за такою формулою:

$$z_i = \varphi_{\theta'}\left(c_i, u_{i-1}, z_{i-1}\right), \tag{3.4}$$

де φ_θ – функція, параметризована θ, яка приймає в якості параметру новий символ c_i і u_{i-1}, z_{i-1} до $(i-1)$-го символу.

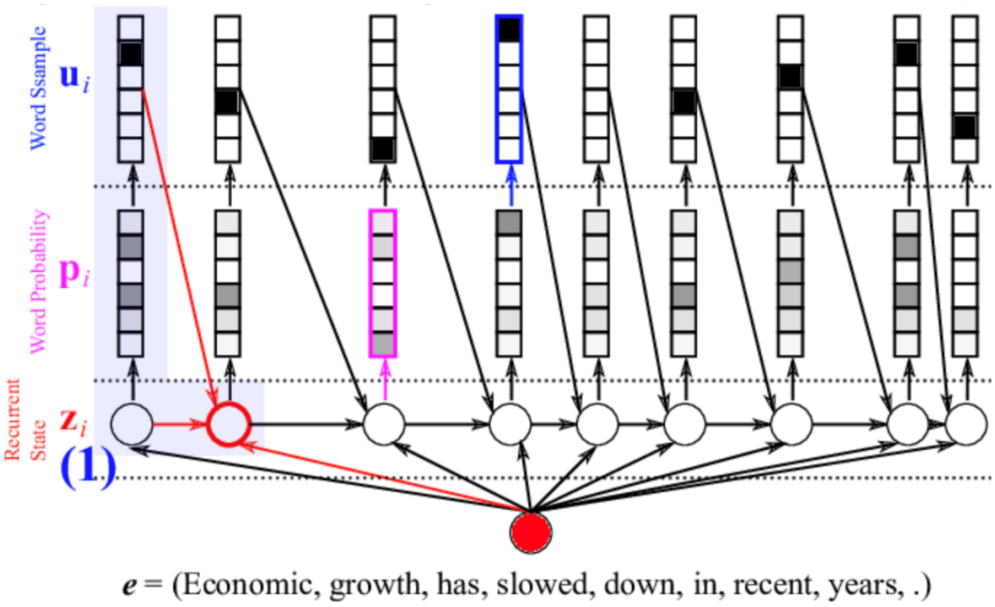

Рис. 3.13. Ілюстрація процесу обчислення внутрішнього стану РНМ z_i

На рис. 3.13 показаний процес обчислення. Отримавши прихований внутрішній стан декодера $\overrightarrow{z_i}$, оцінено кожне цільове слово, на основі того, наскільки ймовірно його поява після всіх попередніх проаналізованих слів при даному вихідному реченні. Для цього необхідно присвоїти ймовірність p_i кожному слову (див. рис. 3.14). Слід врахувати і те, що, ймовірність відрізняється від оцінки тим, що сума ймовірностей всіх можливих слів дорівнює одиниці, а сума оцінок – не має.

Спочатку обчислено оцінку для кожного слова k на основі прихованого стану $\overrightarrow{z_i}$ наступним чином:

$$e(k)=\overrightarrow{w_k^T}\overrightarrow{z_i}+b_k\ , \tag{3.5}$$

де $\overrightarrow{w_k}$ – вектор цільового слова, а b_k – зміщення.

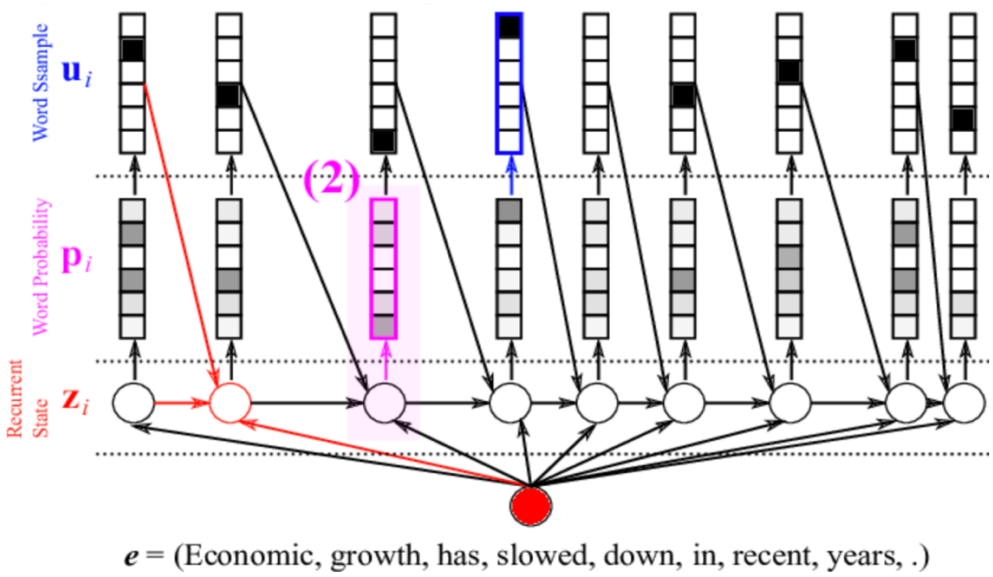

Рис. 3.14. Зображення процесу присвоєння ймовірність p_i слову

Розглянемо перший доданок – скалярний добуток двох векторів. Скалярний добуток є більшим, коли вектор цільового слова $\overrightarrow{w_k}$ і внутрішній стан $\overrightarrow{z_i}$ декодера подібні один одному, і в іншому випадку менше. Слід пам'ятати: скалярний добуток дає довжину проекції одного вектора на інший; якщо вони подібні вектори (майже паралельні), то проекція довша, ніж якщо вони дуже різні (майже перпендикулярні).

Таким чином, цей механізм має високе значення, якщо він добре узгоджується з внутрішнім станом декодера.

Після того, як обчислено оцінки для кожного слова, перетворено їх у відповідні ймовірності наступним чином:

$$p(w_i=k|w_1,w_2,...,w_{i-1},c_i)=\frac{\exp\left(e\left(k\right)\right)}{\Sigma_j \exp(e(j))}, \quad (3.6)$$

Саме цей тип нормалізації називається софтмакс (softmax).

Тепер є розподіл ймовірностей для цільових слів, яке можна використовувати, щоб вибрати слово шляхом вибирання або відбору вибірки (англ. sampling) розподілу, як показано на рис. 3.15.

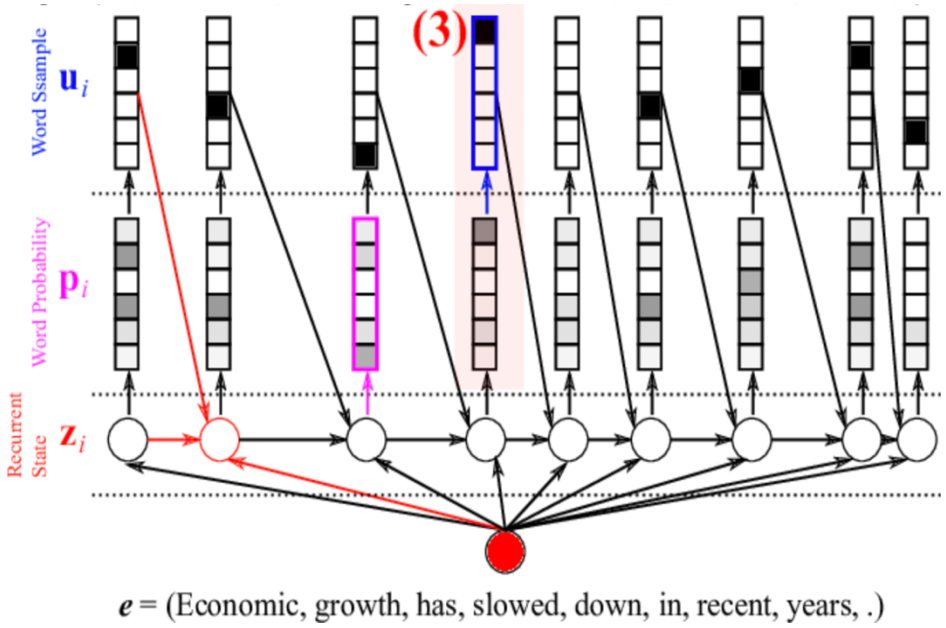

Рис. 3.15. Ілюстрація вибору наступного слова

Вибравши *i*-е слово, повертаємося до першого кроку і обчислюємо прихований внутрішній стан декодера (див. рис. 3.13), оцінюємо і нормалізуємо цільові слова (див. рис. 3.14) і вибираємо наступне $(i+1)$-е слово (див. рис. 3.15). Процес повторюється, поки не буде досягнутий кінець пропозиції (<eos>).

Також слід згадати про Beam Search або променевий пошук, що дозволяє перевірити колекцію найкращих підказок із поточної моделі, а не лише першу без врахування інших можливих варіантів. В інформатиці, променевий пошук – це евристичний алгоритм пошуку, що досліджує граф, розширюючи найперспективніші вузли в обмеженому їх наборі. Він є оптимізацією так званого «пошуку найпершого ліпшого», що суттєво знижує його вимоги до необхідної кількості пам’яті.

Таким чином, поєднання зазначених технологій для роботи з текстовими даними дає можливість якісного навчання моделі. У даному розділі приведений алгоритм використання даних технік для досягнення найкращого результату при введенні тексту за критерієм повноти пропонованих варіантів підказок.

4. ОПИС РЕАЛІЗАЦІЇ ПРОГРАМНИХ ЗАСОБІВ

4.1. Опис структур даних

Процес реалізації програмного забезпечення для предиктивного вводу має за основу створення нейронної мережі для побудови системи розумного доповнення тексту. Оскільки кінцевою метою розроблення є досягнення найкращого результату при введенні тексту в браузері за критерієм повноти пропонованих варіантів підказок, то виникає необхідність у створенні відповідної бази даних текстових елементів для навчання моделі нейронної мережі.

Запропоновано використання даних із сервісу Reddit [37]. Після завантаження повного пакету даних коментарів та відповідей на них із зазначеного сайту з’явилася велика кількість зайвої інформації. Тому виникла необхідність у виокремленні корисних текстових елементів для поставленої задачі створення бази даних для навчання моделі.

Для початку за допомогою SQL-запиту створено таблицю наступного вигляду та заповнено її даними з сервісу Reddit (див. рис. 4.1).

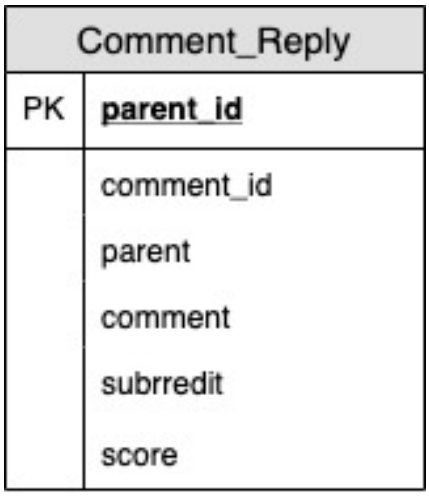

Рис. 4.1. Зображення структури таблиці бази даних для навчання моделі нейронної мережі

Відповідно до рисунку маємо таке відношення між елементами таблиці:

- parent_id – id коментаря, який є батьківським для інших, тобто для нього існують коментарі;
- comment_id – id коментаря під батьківським коментарем;
- parent – батьківський коментар, тобто вміст коментаря, на який існують відповіді;
- comment – вміст відповіді на батьківський коментар;
- subreddit – ім’я автора коментаря;
- score – позиція відносно інших коментарів у сервісі Reddit.

Також сформовано пари коментарів (див. рис. 4.2) зі структурою подібною до «питання-відповідь» структури, де замість питання – батьківський коментар, а у ролі відповіді – відповідь на даний коментар.

```
Total rows read: 10000, Paired rows: 2, Time: 2019-05-21 21:38:16.382303
Total rows read: 20000, Paired rows: 12, Time: 2019-05-21 21:38:27.936030
Total rows read: 30000, Paired rows: 23, Time: 2019-05-21 21:38:39.471389
Total rows read: 40000, Paired rows: 37, Time: 2019-05-21 21:38:51.625061
```

Рис. 4.2. Ілюстрація процесу формування пар коментарів

Отримані дані занесено до бази, представленої на рис. 4.3.

	parent_id	comment_id	parent	comment	subreddit	score
1	t1_cnasass	t1_cnasb93	[It's evidence th…	[Image](http://imgs…	videos	2
2	t1_cnas9iu	t1_cnasbbd	Forgetting song l…	Following this, I h…	AskReddit	21
3	t1_cnas9bl	t1_cnasbgm	Truth doesn't mat…	I will always laugh…	KotakuInAction	8
4	t1_cnas9rd	t1_cnasbla	WHO IS GILBERT?	AAA	Omnipotent_L…	3
5	t1_cnasa0x	t1_cnasbm9	Wearing my Kyrie …	With our combined l…	clevelandcavs	2
6	t1_cnasazf	t1_cnasbno	Cons: injuries pr…	Hit the nail on the…	nfl	3
7	t1_cnasaxr	t1_cnasbrb	Yep	#**Wooooooooooooooo…	ProtectAndSe…	7
8	t1_cnas9y3	t1_cnasbs0	They could put on…	Beautiful.	TumblrInAction	5
9	t1_cnas9g3	t1_cnasbsv	The roaring 20's …	F. Scott Fitzgerald…	AskReddit	2
10	t1_cnas9nd	t1_cnasbui	It wasn't on Marr…	Absolutely. If anyt…	nfl	9
11	t1_cnasarj	t1_cnasbym	You can come to m…	ill be there!	teenagers	2
12	t1_cnas9so	t1_cnasbzd	Jimmy Butler isn'…	My unpopular (maybe…	nba	77
13	t1_cnas9qa	t1_cnasbzs	I urge you to est…	I'm 17 and I live w…	atheism	9
14	t1_cnasans	t1_cnasc0q	I have a list of …	haha, damn! i defi…	pics	2
15	t1_cnasagz	t1_cnasc2h	Hope you didn't f…	Portfolio definitel…	leagueoflege…	11
16	t1_cnasa8h	t1_cnasc2p	What if it is ava…	Yeah I get what you…	GrandTheftAu…	2
17	t1_cnasasg	t1_cnasc5v	*HAPPY NEW YEAR, …	It's new year now. …	BloodWorld	3
18	t1_cnasavn	t1_cnasc8s	Things you can sa…	Quit fucking me so …	nba	103

Рис. 4.3. Зображення бази даних для навчання моделі нейронної мережі

Таким чином, у кінцевому результаті формування даних для навчання моделі нейронної мережі отримано два файли «train.from», «train.to». Відповідно до алгоритму, обраного для вирішення поставленого завдання створення системи предиктивного вводу, вміст зазначених файлів співставляється у пару певного рядка.

Відбувається процес навчання моделі аналогічно до процесу при використанні базової seq2seq архітектури при перекладі, але замість фрази іншою мовою маємо відповідь на неї.

4.2. Модуль представлення даних

Модуль для представлення даних при взаємодії з користувачем у веб-браузері являє собою розширення для веб-браузеру Chrome. Це маленька програма, яка модифікує і доповнює функціональність браузера Google Chrome. Для створення повноцінного розширення необхідно мати знання HTML, CSS, JavaScript.

Після написання, файли пакуються в спеціальний файл з розширенням .crx, який собою являє zip архів (див. рис. 4.4). У такому вигляді користувач зможе встановити розширення.

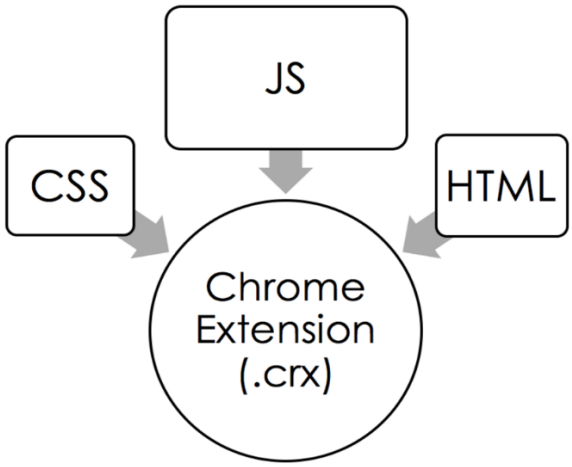

Рис. 4.4. Ілюстрація складових елементів zip архіву
chrome розширення

Перевагою такої реалізації є те, що за рахунок того що цей пакет містить всі необхідні файли, chrome розширення не залежить від ресурсів з інтернету і здатний коректно працювати навіть в оффлайн режимі.

Загалом структура розширення даного браузера виглядає так, як представлено на рис. 4.5. Оскільки компоненти такої структури не усі є обов'язковими, тому структура залежить від особливостей реалізації та конкретних потреб.

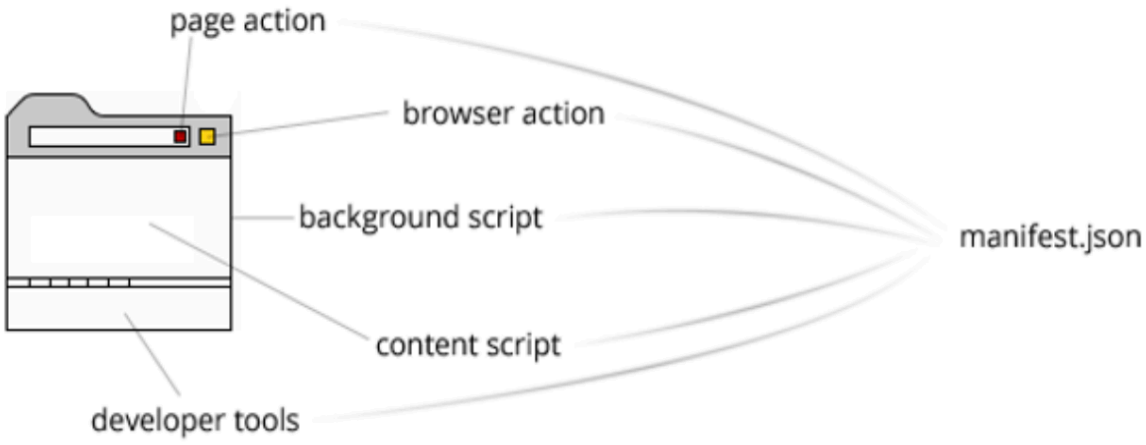

Рис. 4.5. Зображення структури розширення chrome розширення

Розглянуто основні елементи зазначеної вище структури більш детально.

4.2.1. Manifest.json

Manifest.json – головний файл chrome розширення. Тут міститься інформація про доступи які потрібні, про файли, які підключаються, про налаштування безпеки та інше. Файл містить наступні позиції:

- manifest_version – версія маніфест файлу;
- name – назва розширення;

- description – опис розширення;
- version – версія розширення;
- permissions – масив з назвами доступів, які необхідні для коректної роботи розширення;
- content_scripts – масив файлів, які будуть підключені як контент скрипти;
- background – опис файлу або файлів, які будуть виконувати роль background скрипта і сторінки;
- web_accessible_resources – набір файлів, які мають відкритий доступ ззовні;
- browser_action – настройка відповідної кнопки, в панелі інструментів;
- icons – списки іконок за стандартними розмірами 16 48 і 128.

4.2.2. Background page

Також є одним з головних елементів розширення веб-браузеру. Відповідно до офіційної документації [38], даний елемент являє собою невидиму сторінку, яка містить основну логіку розширення. Також слід відмітити, що Background page запускається та виконується у фоновому режимі, під час запуску браузеру та міститься в оперативній пам’яті як фоновий процес протягом сесії.

Для оптимізації використання ресурсів була розроблена концепція сторінок подій (англ. Event Pages). Вони виконують усі ті ж функції, що і Background page, але спрямовані на вирішення проблеми продуктивності та ірраціонального використання ресурсів. Головною відмінністю такого підходу є те, що замість безперервної роботи у фоновому режимі, сторінка події запускається тільки тоді, коли це необхідно, наприклад, оброблення конкретної події. Такий підхід дозволяє звільнити пам’ять до того моменту поки конкретна подія не спрацює наступного разу. З точки зору програмної

реалізації все залишається незмінним, окрім значення властивості persistent у файлі manifest.json.

4.3. Синхронізація обміну даними між об’єктами системи

З огляду на результати проведеного аналізу мов програмування для розроблення програмного забезпечення предиктивного вводу, маємо дві позиції: для оброблення та генерування даних мова програмування Python, для веб-розширення – Javascript. У такому випадку виникає питання синхронізації між модулями програмного продукту.

Оскільки скрипти вмісту виконуються в контексті веб-сторінки, а не розширення, вони часто потребують певного способу спілкування з рештою частиною розширення. Зв'язок між розширеннями та їхніми скриптами вмісту працює за допомогою передачі повідомлень. Кожна зі сторін може слухати повідомлення, надіслані з іншого кінця, і відповідати на той самий канал.

Повідомлення може містити будь-який дійсний об'єкт JSON (null, boolean, number, string, array або object). Існує простий API для одноразових запитів і більш складний API, що дозволяє мати довговічні з'єднання для обміну кількома повідомленнями із загальним контекстом. Можна також надіслати повідомлення на інше розширення, якщо відомо його ідентифікатор, який описаний у розділі повідомлень про перехресне розширення.

У реалізації веб-розширення для дипломного проекту потрібно надіслати лише одне повідомлення іншій частині нашого розширення (і додатково отримати відповідь). Тому було використано спрощений runtime.sendMessage або tabs.sendMessage. Це дозволяє відправити одноразове повідомлення JSON-serializable зі сценарію вмісту до розширення або навпаки, відповідно.

На іншому кінці, де чекають на повідомлення, налаштовано слухач подій runtime.onMessage для обробки повідомлення. При цьому також слід

враховувати, якщо для подій onMessage прослуховуються кілька сторінок, то лише перший, хто викликає sendResponse () для певної події, успішно надішле відповідь. Усі інші відповіді на цю подію будуть проігноровані.

Що стосується даних, якими оперує програмне забезпечення під час взаємодії з користувачем, то вони обробляються та генеруються на стороні сервера. Відомо, що HTTP – це протокол для веб-сайтів. Інтернет використовує його для взаємодії та спілкування з комп'ютерами та серверами. Приклади його використання зустрічаються щодня: при введенні ім'я веб-сайту в адресний рядок браузера, на сервер надсилаються дані за допомогою HTTP-запиту; при переході до адресного рядка і введенні google.com на сервер Google також надсилається HTTP-запит і у свою чергу сервер отримує запит і з'ясовує, як саме інтерпретувати цей запит. Для нашого завдання синхронізації сервера та клієнтської частини використано Flask – фреймворк для створення веб-застосунків мовою програмування Python. Він надає можливість вільного обміну даними між навченою раніше моделлю нейронної мережі та клієнтською частиною. Таким чином, зображено потік даних програмного забезпечення предиктивного вводу тексту в браузері (див. Додаток 1).

4.4. Опис інтерфейсу користувача

Коротко описано процес взаємодії користувача із системою та наведено декілька зображень інтерфейсу користувача. Оскільки розроблюваний програмний продукт орієнтований на використання у веб-браузерах, тому проілюстровано загальний процес взаємодії із системою предиктивного вводу. Для графічного представлення зображено діаграму діяльності (див. Додаток 1).

Після активації розширення веб-браузеру (див. Керівництво користувача, розділ 1) користувачеві стають доступні його функції, а саме автоматичне доповнення тексту при відповіді на отримане повідомлення (див. Додаток 1).

При надходженні повідомлення, текст відправляється до навченої попередньо моделі нейронної мережі та аналізується. У результаті формуються можливі відповіді на отриманий текст, які відображаються у полі вводу.

4.5. Тестові випадки

Відповідно до вимог, сформованих для програмного забезпечення предиктивного введення тексту в браузері, сформовано тестові випадки, а саме сценарії димового тестування (див. табл. 4.1).

Таблиця 4.1

Тестові випадки

№	Вхідні параметри	Дія	Очікуваний результат
1	Неактивоване розширення для веб-браузеру	Натискання на значок розширення	Активація розширення для веб-браузеру
2	Неактивоване розширення для веб-браузеру	Написання повідолення	Поле вводу залишається незмінним. Написання повідомлення без предиктивного вводу тексту.
3	Активоване розширення для веб-браузеру	Написання повідомлення	При введені у полі вводу перших символів з’являється текст підказки більш світлого відтінку.
4	Активоване розширення для веб-браузеру	Натискання кнопки «Enter» при підставленій підказці	Відправлення повідомлення із відповідним вмістом

Продовження табл. 4.1

№	Вхідні параметри	Дія	Очікуваний результат
5	Активоване розширення для веб-браузеру	Натискання кнопки «стрілка вправо» при існуючій підказці	Підстановка можливої відповіді для відправки повідомлення
6	Активоване розширення для веб-браузеру	Натискання будь-якої, окрім кнопки «стрілка вправо» та «Enter» при існуючій підказці	Зміна положення курсору (у випадку натиснення стрілочних кнопок), зміна введеного тексту (у випадку натиснення інших кнопок)
7	Активоване розширення для веб-браузеру	Написання повідомлення: ввід тексту підказки, довжина якої більша за довжину поля вводу	Перехід тексту на новий рядок
8	Активоване розширення для веб-браузеру	Написання повідомлення: підказка не сформована	Поле вводу залишається незмінним, написання повідомлення без предиктивного вводу.

4.6. Шляхи подальшого розвитку проекту

Після аналізу результатів розробленого програмного забезпечення для предиктивного введення тексту браузері, сформовано шляхи подальшого розвитку проекту.

У першу чергу необхідно:

1. Використання оптимізованого алгоритму групування лексем для швидкого утворення бази слів.
2. Впровадження паралелізму з використанням workers та shared array buffer в JavaScript.

3. Інтеграція з іншими браузерами та мобільними пристроями.
4. Додавання можливості застосування предиктивного введення при використанні різних мов.
5. Формування словника користувача.
6. Додавання можливості налаштування системи на повсякденну та бізнес бесіду.
7. Дослідження можливості одночасного застосування Rule-based та ML підходів для реалізації більш швидкої та коректної роботи системи.

ВИСНОВКИ

Метою даного дипломного проекту було створення зручної системи обміну інформацією за рахунок розроблення програмного забезпечення для предиктивного введення тексту в браузері. Даний програмний продукт реалізовано та впроваджено у використання в браузері Chrome.

Аналіз існуючих програмних рішень, попередньо виконаний в дипломному проекті, показав доцільність створення даного проекту. Для розроблення були використані відповідні технології, а саме: мова програмування Python для реалізації модулю генерування та обробки даних на основі нейронної мережі та глибокого навчання та мова програмування JavaScript для створення веб-розширення для браузеру Chrome та відповідний інтерфейс. Зазначені зароби реалізації дозволили забезпечити стабільну роботу системи.

Розроблене програмне забезпечення:

1. Дозволяє налаштовувати веб-розширення на роботу;
2. Має розроблені алгоритми для пошуку поля вводу;
3. Забезпечує динамічну появу підказок, сформованих системою предиктивного вводу;
4. Має візуалізацію дій користувача при взаємодії.

Особливу увагу під час розроблення даного програмного продукту було приділено формуванню тексту підказок.

Розроблення виконано у повному обсязі, відповідає всім вимогам, описаних в документі, тестування продукту виконано у відповідності до затвердженої програми та методики тестування.

Використання розробленого продукту дозволить досягти найкращого результату при введенні тексту у браузері за критерієм повноти пропонованих варіантів підказок.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Nadir Zanini. Text Mining: An introduction to theory and someapplications [Електронний ресурс] / Nadir Zanini, Vikas Dhawan // RESEARCH MATTERS: ISSUE 19. – 2015. – Режим доступу до ресурсу: https://www.researchgate.net/publication/304140500_Text_Mining_An_introduction_to_theory_and_some_applications(https://www.researchgate.net/publication/304140500_Text_Mining_An_introduction_to_theory_and_some_applications).
2. QuickType [Електронний ресурс] // Apple Inc. All rights reserved. – 2019. – Режим доступу до ресурсу: https://www.apple.com/my/ios/whats-new/quicktype/.
3. Taylor Martin. The predictive keyboard in Jelly Bean is great, but SwiftKey 3 is better [Електронний ресурс] / Taylor Martin. – 2012. – Режим доступу до ресурсу: https://www.phonedog.com/2012/07/19/the-predictive-keyboard-in-jelly-bean-is-great-but-swiftkey-3-is-better.
4. Василевский Э. Итоги Google I/O 2018: Android P, Google Lens и многое другое [Електронний ресурс] / Эрнест Василевский. – 2018. – Режим доступу до ресурсу: https://hi-news.ru/technology/itogi-google-i-o-2018-android-p-google-lens-i-mnogoe-drugoe.html.
5. Using Quick Replies When Responding to InMail Messages [Електронний ресурс] // LinkedIn Corp. – 2019. – Режим доступу до ресурсу: https://www.linkedin.com/help/linkedin/answer/62002/using-quick-replies-when-responding-to-inmail-messages?lang=en.
6. Adam Fendelman. What is T9 Predictive Text? [Електронний ресурс] / Adam Fendelman. – 2018. – Режим доступу до ресурсу: https://www.lifewire.com/definition-of-t9-predictive-text-578677.
7. John Harrison. Introduction to Functional Programming / John Harrison., 1997. – 168 с.

8. John Harrison. Introduction to Functional Programming / John Harrison., 1997. – 168 с.
9. Richard L. Halterman. Fundamentals of Python Programming / Richard L. Halterman. – Southern Adventist University, 2017. – 669 с.
10. R (programming language) [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/R_(programming_language).
11. S (мова програмування) [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/S_(programming_language).
12. Scheme (programming language) [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Scheme_(programming_language).
13. Dr. Axel Rauschmayer. Speaking JavaScript /Dr. Axel Rauschmayer, 2014. – 460 с.
14. Сарницкий Я. ES6, ES8, ES2017: что такое ECMAScript и чем это отличается от JavaScript [Електронний ресурс] / Ярослав Сарницкий. – 2017. – Режим доступу до ресурсу: https://tproger.ru/translations/wtf-is-ecmascript/.
15. Introduction to Node.js [Електронний ресурс] – Режим доступу до ресурсу: https://nodejs.dev/.
16. Lokesh Gupta. What is Java programming language? [Електронний ресурс] / Lokesh Gupta – Режим доступу до ресурсу: https://howtodoinjava.com/java/basics/what-is-java-programming-language/.
17. MICHAEL COPELAND. What's the Difference Between Artificial Intelligence, Machine Learning, and Deep Learning? [Електронний ресурс] / MICHAEL COPELAND. – 2016. – Режим доступу до ресурсу: https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/.
18. Andy Meng. WHAT IS PINTEREST, AND HOW DOES IT WORK? [Електронний ресурс] / Andy Meng. – 2019. – Режим доступу до ресурсу: https://www.infront.com/blog/the-blog/what-is-pinterest-and-how-does-it-work.

19. Artificial neural network [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Artificial_neural_network.
20. HTML [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/HTML.
21. Cascading Style Sheets [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Cascading_Style_Sheets.
22. Ilya Bodrov-Krukowski. Angular Introduction: What It Is, and Why You Should Use It [Електронний ресурс] / Ilya Bodrov-Krukowski. – 2018. – Режим доступу до ресурсу: https://www.sitepoint.com/angular-introduction/.
23. Scott Morris. What is jQuery, and What is it Used for? [Електронний ресурс] / Scott Morris. – 2019. – Режим доступу до ресурсу: https://skillcrush.com/2018/07/13/what-is-jquery-used-for/.
24. Tania Rascia. What is Bootstrap and How Do I Use It? [Електронний ресурс] / Tania Rascia. – 2015. – Режим доступу до ресурсу: https://www.taniarascia.com/what-is-bootstrap-and-how-do-i-use-it/.
25. Scott Morris. AJAX—What It Is, How It Works, and What It's Used For [Електронний ресурс] / Scott Morris. – 2018. – Режим доступу до ресурсу: https://skillcrush.com/2018/04/26/what-is-ajax/.
26. Ari Lerner. What is React? [Електронний ресурс] / Ari Lerner – Режим доступу до ресурсу: https://www.fullstackreact.com/30-days-of-react/day1/.
27. Scott Morris. Everything You Need to Know About PHP [Електронний ресурс] / Scott Morris. – 2018. – Режим доступу до ресурсу: https://skillcrush.com/2012/04/11/php/.
28. Anna Birukova. 8 лучших PHP Framework для веб-разработчиков [Електронний ресурс] / Anna Birukova. – 2019. – Режим доступу до ресурсу: https://www.hostinger.com.ua/rukovodstva/8-luchshih-php-framework-dla-web-razrabotchikov/.

29. IEEE Standart Glossary of Software Engineering Terminology [Електронний ресурс]. –1990. – Режим доступу до ресурсу: http://www.mit.jyu.fi/ope/kurssit/TIES462/Materiaalit/IEEE_SoftwareEngGlossary.pdf.
30. What are System Requirements Specifications/Software (SRS)? [Електронний ресурс] // Inflectra Corporation. – 2018. – Режим доступу до ресурсу: https://www.inflectra.com/ideas/topic/requirements-definition.aspx.
31. Stakeholder (corporate) [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Stakeholder_(corporate).
32. Mohammed Ma`amari. NLP // Sequence to Sequence Networks// Part 2 // Seq2seq Model (EncoderDecoder Model) [Електронний ресурс] / 31. Mohammed Ma`amari – Режим доступу до ресурсу: https://towardsdatascience.com/nlp-sequence-to-sequence-networks-part-2-seq2seq-model-encoderdecoder-model-6c22e29fd7e1.
33. Mike Schuster. Bidirectional Recurrent Neural Networks [Електронний ресурс] / Mike Schuster, Kuldip K. Paliwal. – 1997. – Режим доступу до ресурсу: https://tproger.ru/translations/neural-network-zoo-2/.
34. Jason Brownlee. How to Implement a Beam Search Decoder for Natural Language Processing [Електронний ресурс] / Jason Brownlee. – 2018. – Режим доступу до ресурсу: https://machinelearningmastery.com/beam-search-decoder-natural-language-processing/.
35. Полыковский Д.А. Механизмы внимания в нейронных сетях / Полыковский Д.А. – Москва, 2017. – 22 с.
36. John S. Bridle. Training Stochastic Model Recognition Algorithms as Networks can Lead to Maximum Mutual Information Estimation of Parameters / John S. Bridle. – UK: Royal Signals and Radar Establishment, 1989. – 211-217 с.
37. Dzmitry Bahdanau. NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE / Dzmitry Bahdanau, Blunsom. – 2015.

38. Jake Widman. What is Reddit? A beginner’s guide to the front page of the internet [Електронний ресурс] / Jake Widman, Will Nicol. – 2019. – Режим доступу до ресурсу: https://www.digitaltrends.com/web/what-is-reddit/.
39. What are extensions? [Електронний ресурс] – Режим доступу до ресурсу: https://developer.chrome.com/extensions/overview.

ДОДАТКИ

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2018 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ПРЕДИКТИВНОГО ВВЕДЕННЯ ТЕКСТУ В БРАУЗЕРІ

Програма та методика тестування

ДП.045440-04-51

"ПОГОДЖЕНО"

Керівник проекту:

_____ Т.М. Заболотня

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ Л.О. Довганюк

2018

ЗМІСТ

1. ОБ’ЄКТ ВИПРОБУВАНЬ

Програмне забезпечення для предиктивного введення тексту в браузері, яке представлено у вигляді розширення для веб-браузеру Chrome.

2. МЕТА ТЕСТУВАННЯ

У процесі тестування має бути перевірено наступне:

1. Можливість налаштування системи веб-розширення.
2. Забезпечення коректної роботи при взаємодії з користувачем.
3. Забезпечення динамічного представлення підказок, сформованих системою.
4. Наявність інтуїтивно-зрозумілого інтерфейсу.

3. МЕТОДИ ТЕСТУВАННЯ

Тестування виконується відповідно до технік Sanity Testing. Тобто з використанням базового тесту на перевірку працездатності програмного проекту.

Використовуються наступні методи:

1. Функціональне тестування, зокрема на рівні UI testing (перевірка інтерфейсу на предмет коректності роботи).
2. Нефункціональне тестування, зокрема на рівні Reability (тестування надійності) та Efficienty testing (тестування ефективності).

4. ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Працездатність програмного продукту перевіряється шляхом:

1. Динамічного ручного тестування – введенням граничних та недопустимих значень в поля, які доступні для використання у рамках даного програмного проекту.
2. Динамічного ручного тестування на відповідність функціональним вимогам.
3. Статичного тестування коду.
4. Тестування програмного застосунку у веб-браузері Chrome.
5. Тестування при максимальному навантаженні.
6. Тестування стабільності роботі при різних умовах.
7. Тестування зручності використання.
8. Тестування інтерфейсу.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2019 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ПРЕДИКТИВНОГО ВВЕДЕННЯ ТЕКСТУ В БРАУЗЕРІ

Керівництво користувача

ДП.045440-05-34

"ПОГОДЖЕНО"

Керівник проекту:

_____ Т.М. Заболотня

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ Л.О.Довганюк

2019

ЗМІСТ

1. Опис структури програмного застосунку

Програмне забезпечення для предиктивного введення тексту в браузері являє собою розширення для веб-браузеру Chrome. Як і звичні користувачеві веб-розширення, розроблений програмний продукт має значок у верхньому правому куті сторінки веб-браузера, який дозволяє увімкнути систему предиктивного введення.

При активації розробленого веб-розширення вміст відкритих користувачем сторінок не змінюється. Доповнюються лише дані у полі вводу.

2. Опис процедури налаштування системи для роботи

Процес налаштування програмного забезпечення предиктивного введення тексту в браузері являє собою активацію системи при натисненні відповідної кнопки у верхньому правому куті сторінки веб-браузера на панелі інструментів (див. рис. 2.1).

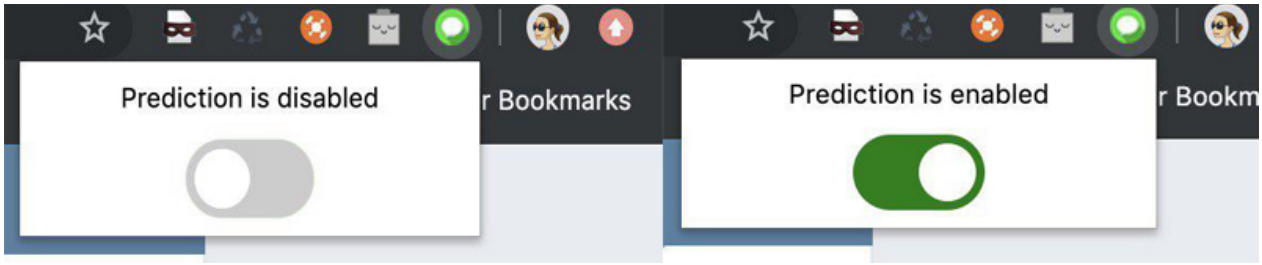

Рис. 2.1. Ілюстрація процесу активації роботи системи предиктивного введення тексту в браузері

3. Користування системою предиктивного введення

Коротко опишемо процес взаємодії користувача із системою та наведемо декілька зображень інтерфейсу користувача. Оскільки розроблюваний програмний продукт орієнтований на використання у веб-браузерах, тоді опишемо загальний процес взаємодії із системою предиктивного вводу.

Після активації розширення веб-браузеру користувачеві стають доступні його функції, а саме автоматичне доповнення тексту при відповіді на отримане повідомлення. Візуальне представлення відкритої сторінки веб-браузера при цьому не змінюється.

При надходженні повідомлення, текст обробляється та аналізується. У результаті формуються можливі відповіді на отриманий текст, які відображаються у полі вводу відкритої сторінки більш світлим відтінком кольору при введені перших символів користувачем (див. рис. 3.1). Для підстановки отриманих від програмного застосунку даних необхідно натиснути кнопку «стрілка вправо», після чого можна відправити повідомлення. У разі, якщо ж користувач не бажає використовувати запропоновані підказки або ж система не надала їх, тоді відбувається звичний процес відправлення повідомлення користувачем вручну.

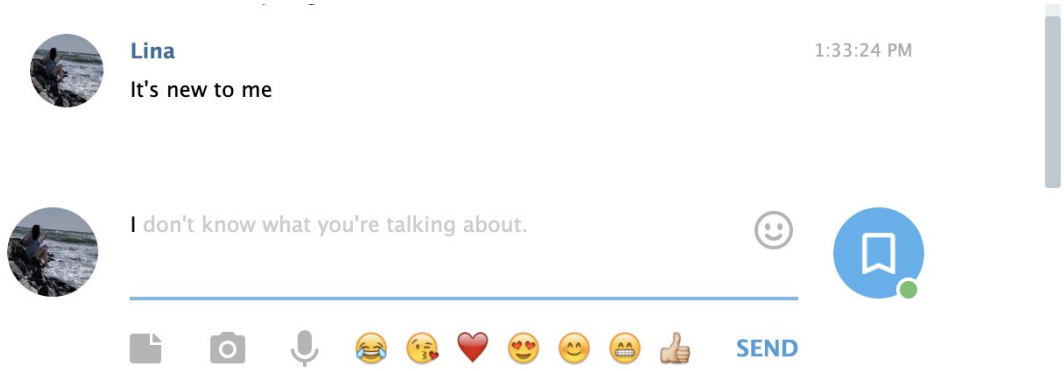

Рис. 3.1. Зображення підказок, сформованих системою предиктивного вводу тексту у веб-браузері Chrome